Software Architectural Transformation: A New Approach to Low Energy Embedded Software

T.K. Tan, A. Raghunathan, and N.K. Jha

*Design, Automation, and Test in Europe (DATE) 2003*

2008. 5. 7

AhRim Han

© KAIST SE LAB 2008
Contents

- Introduction
- Overview of the approach
- Energy minimization methodology
  - SAG & OS energy macro-models
  - Software architectural transformations
- Case study
- Related work
- Conclusion
- Our study: Model-based power estimation
Introduction (1/2)

- Low power techniques at various levels of design abstraction

- High abstraction level
  - Power saving is more efficient and power estimation is faster

- Software domain
  - Previous studies only focused in instruction-level and compiler optimizations
Factors that affect power consumption in a software architecture level

- Software architectural style selection
- Software architecture design: components & connectors

In this paper,

- Provide a systematic methodology for applying software architectural transformations to derive an power-optimized architecture for given embedded software
  - Architecture style - OS-driven multi-process architectural style
  - Components - application processes, signal handlers, device drivers, etc.
  - Connectors - inter-process communication (IPC), synchronization mechanisms
Overview of the approach

- **Software architectural level energy minimization methodology**

1. Constructing a software architecture graph (SAG) to represent a program

2. Deriving an initial profile of the energy consumption and execution statistics using a energy simulation framework

3. Evaluating the energy impact of atomic **software architecture transformations** through the use of energy macro-models

4. Constructing sequences of atomic transformations that result in maximal energy reduction

5. Generation of a program source code to reflect the optimized software architecture

© KAIST SE LAB 2008
Software architecture graph (SAG)

SAG for a situational awareness system

- Hardware device
- Active hardware device
- Device drivers
- Association
- Application processes
- Signal handlers

IPC mechanism

Communication of data or control messages
- Blocking communication
  - if none,
  - Non-blocking

© KAIST SE LAB 2008
**Definition**

- A function expressing the relationship between
  - Energy consumption of a system function and predefined parameters

**Example**

- Energy macro-models in Linux

<table>
<thead>
<tr>
<th>Sub-systems</th>
<th>System Functions</th>
<th>Macro-models (nJ)</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process Manager</td>
<td>fork()</td>
<td>$E_{parent} = 30540$</td>
<td>$\epsilon = 3.6%$</td>
</tr>
<tr>
<td>Process Manager</td>
<td>fork()</td>
<td>$E_{child} = 4300$</td>
<td>$\epsilon = 0.1%$</td>
</tr>
<tr>
<td>Process Manager</td>
<td>getpid()</td>
<td>$E = 177$</td>
<td>$\epsilon = 0.1%$</td>
</tr>
<tr>
<td>IPC</td>
<td>msgsnd(x bytes)</td>
<td>$E = 4752 + 1.08x$</td>
<td>$\epsilon = 1.4%$</td>
</tr>
<tr>
<td>IPC</td>
<td>msgrcv(x bytes)</td>
<td>$E = 4913 + 1.19x$</td>
<td>$\epsilon = 1.3%$</td>
</tr>
<tr>
<td>IPC</td>
<td>pipe write(x bytes)</td>
<td>$E = 1147 + 1.64x$</td>
<td>$\epsilon = 11.2%$</td>
</tr>
<tr>
<td>IPC</td>
<td>pipe read(x bytes)</td>
<td>$E = 690 + 1.65x$</td>
<td>$\epsilon = 14.4%$</td>
</tr>
<tr>
<td>IPC</td>
<td>signal()</td>
<td>$E = 2739.9$</td>
<td>$\epsilon = 0.1%$</td>
</tr>
<tr>
<td>Timer Functions</td>
<td>gettimeofday()</td>
<td>$E = 495$</td>
<td>$\epsilon = 0.1%$</td>
</tr>
<tr>
<td>Timer Functions</td>
<td>settimeofday()</td>
<td>$E = 595$</td>
<td>$\epsilon = 0.1%$</td>
</tr>
<tr>
<td>Memory Manager</td>
<td>calloc(x blocks, size y)</td>
<td>$E = 287 + 0.986xy$</td>
<td>$\epsilon = 5.1%$</td>
</tr>
<tr>
<td>Memory Manager</td>
<td>malloc()</td>
<td>$E = 123$</td>
<td>$\epsilon = 2.0%$</td>
</tr>
<tr>
<td>Memory Manager</td>
<td>free()</td>
<td>$E = 160$</td>
<td>$\epsilon = 27.5%$</td>
</tr>
<tr>
<td>Memory Manager</td>
<td>memcpy(x bytes)</td>
<td>$E = 110 + 0.98x$</td>
<td>$\epsilon = 4.6%$</td>
</tr>
<tr>
<td>File System</td>
<td>open()</td>
<td>$E = 2351$</td>
<td>$\epsilon = 2.1%$</td>
</tr>
<tr>
<td>File System</td>
<td>close()</td>
<td>$E = 690$</td>
<td>$\epsilon = 0.1%$</td>
</tr>
<tr>
<td>File System</td>
<td>file read(x bytes)</td>
<td>$E = 1209 + 1.06x$</td>
<td>$\epsilon = 16.8%$</td>
</tr>
<tr>
<td>File System</td>
<td>file write(x bytes)</td>
<td>$E = 2430 + 2.75x$</td>
<td>$\epsilon = 15.7%$</td>
</tr>
</tbody>
</table>

- Show 5.9% average error with respect to the energy data used to obtain the macro-models
How to make

- Use a regression technique
  - Ex) Energy of a system function `msgsnd()`
    
    \[
    E = c_1 + c_2 x \text{ (} x \text{ – size of bytes for message sending)}
    \]

\[
\begin{pmatrix}
1 & x_1 \\
1 & x_2 \\
. & . \\
1 & x_n
\end{pmatrix}
\begin{pmatrix}
c_1 \\
c_2
\end{pmatrix}
= 
\begin{pmatrix}
E_1 \\
E_2 \\
. \\
E_n
\end{pmatrix}
\]
Where to be used

- Evaluating energy impact of software architectural transformations
  - Configuration (C) of a software architecture is transformed
    - $C_0 \rightarrow C_1 \rightarrow C_2 \rightarrow \ldots \rightarrow C_{\text{optimal}}$
  - It takes too much time to obtain an energy of each $C$ by a detailed system simulation framework
    - $E(C_0), E(C_1), E(C_2), \ldots, E(C_{\text{optimal}})$
  - The effect of software architectural transformations is to alter the manner in which OS services are employed
  - *Energy change estimates* can be obtained by OS macro-models
  - Energy of a transformed software architecture can be evaluated
    - $E(C_0)$ by performing a detailed energy profiling one time
    - $E(C_1) = E(C_0) + \triangle E(T_{C_0 \rightarrow C_1})$, $E(C_2) = E(C_1) + \triangle E(T_{C_1 \rightarrow C_2})$, \ldots,
    - $E(C_{\text{optimal}}) = E(C_{\text{optimal pre}}) + \triangle E(T_{C_{\text{optimal pre}} \rightarrow \text{Optimal}}$
Where to be used (Cont’d)

- Calculating energy change estimates
  - Ex) Amount of energy change incurred by replacing IPC\textsubscript{1} with IPC\textsubscript{2}

- Energy macro models for IPC\textsubscript{1} and IPC\textsubscript{2}
  \[ E_{ipc1}(x) = c_{11} + c_{12}x \]
  \[ E_{ipc2}(x) = c_{21} + c_{22}x \]

- Amount of energy change estimates
  \[ \Delta E = [E_{ipc2}(x) - E_{ipc1}(x)] N_{ipc} \]
  \[ = [(c_{21} - c_{11}) + (c_{22} - c_{12}) x] N_{ipc} \]
Software architectural transformation (1/3)

- Temporal cohesion driven process merging
  \[ \Delta E = -E_{ctx} N_{ctx} \]

- Sequential cohesion driven process merging
  \[ \Delta E = -E_{ipc}(x) N_{ipc} - E_{ctx} N_{ctx} \]

- Temporal cohesion driven IPC merging
  \[ \Delta E = -E_{ipc\_read}(x) N_{ipc\_read} \]
Software architectural transformation (2/3)

- **Intra-process computation migration**

\[ \Delta E = - (E_{ipc\_wr}(x) + E_{ipc\_wr}(y) - E_{ipc\_wr}(x + y)) N_{ipc\_wr} \]

- **Process embedding as signal handlers**

\[ \Delta E = -E_{ctx} N_{ctx} + E_{sig} N_{sig} \]

© KAIST SE LAB 2008
Software architectural transformation (3/3)

- IPC replacement

\[
\Delta E = [E_{ipc2}(x) - E_{ipc1}(x)] N_{ipc}
\]

- System function replacement

\[
\Delta E = [E_{sys2} - E_{sys1}] N_{sys}
\]
Case study (1/3)

- **Experiment environment**
  - Embedded Linux OS: arm-linux v2.2.2
  - Hardware platform: StrongARM embedded processor
  - Energy simulation framework: EMSIM [1]
  - High-level energy macro-models of the Linux OS: [2]

- **Studied various multi-process example programs**

<table>
<thead>
<tr>
<th>Program name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aware</td>
<td>Situational awareness system illustrated in the previous example</td>
</tr>
<tr>
<td>Headphone</td>
<td>Program used in an audio headset application</td>
</tr>
<tr>
<td>Vcam</td>
<td>Embedded software from a video camera recorder</td>
</tr>
<tr>
<td>Climate</td>
<td>Application for collecting and processing climate information</td>
</tr>
<tr>
<td>Navigator</td>
<td>Software from a global navigation</td>
</tr>
<tr>
<td>ATR</td>
<td>Part of an automatic target recognition program</td>
</tr>
</tbody>
</table>

Energy minimization: ex) climate system

Initial SAG
$E_0 = 0.239\text{mJ}$

Final SAG
$E_0 = 0.081\text{mJ}$

% Energy reduction = 66.1%
Results of energy minimization

<table>
<thead>
<tr>
<th>Examples</th>
<th>Original</th>
<th>Optimized</th>
<th>% Energy reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Energy (mJ)</td>
<td># proc</td>
<td># ipc</td>
</tr>
<tr>
<td>Aware</td>
<td>12.956</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>Headphone</td>
<td>1.668</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>Vcam</td>
<td>1.572</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Climate</td>
<td>0.239</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Navigator</td>
<td>1.659</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>ATR</td>
<td>6.940</td>
<td>4</td>
<td>7</td>
</tr>
</tbody>
</table>

Software architecture level techniques for energy reduction can significantly extend and complement existing low power software design techniques!!!
Related work

- Energy minimization techniques at various level
  - Software design
    - Instruction
      - Instruction selection, register allocation, instruction scheduling (e.g., loop optimization by merging and unrolling), etc.
    - Program code
      - Hot operations (e.g., external memory R/W, some coprocessor operations) reduction, data representation change, etc.
  - Hardware design
    - Transistor/Logic
      - Capacitance minimization/low leakage, low voltage, etc.
    - Architecture/System
      - Resource scheduling (e.g., parallel processing, pipelining, shut down)
Conclusion

- Provide a method to design a low power embedded software by software architectural transformations
  - Constructing a software architectural graph (SAG)
  - Deriving initial energy statistics using a detailed energy simulation framework
  - Constructing sequences of atomic software architectural transformations guided by energy change estimates derived from OS energy macro models
Model-based power estimation (1/4)

- Power estimation point in the embedded system development process

```
+-----------------------+                        +-----------------------+                        +-----------------------+
| Requirement           |                        | Power Estimation       |                        | HW Requirement         |
| Specification         |                        |                         |                        | Analysis               |
|                       | feedback               | feedback                | feedback               |                        |
|-----------------------+------------------------+-------------------------+------------------------+------------------------|
| SW Requirement        |                        | HW/SW Design            |                        | HW Design              |
| Analysis              | feedback               |                         | feedback               |                        |
| SW Design             |                        | HW/SW Design Integration| feedback               |                        |
|                       |                       |                         |                        |                       |
| SW Implementation     | feedback               | Virtual Testing and     | feedback               | HW Implementation      |
| and Debugging         |                        | Verification            |                        | and Debugging          |
|                       | feedback               |                         | feedback               |                        |
|                       |                        | Target Installation     |                        |                        |
|                       |                        | Testing and Verification|                        |                        |
|                       |                        |                         | feedback               |                        |
```

© KAIST SE LAB 2008
## Model-based power estimation (2/4)

### Energy estimation studies at various level

<table>
<thead>
<tr>
<th>Type</th>
<th>Level</th>
<th>Author</th>
<th>Method</th>
<th>Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW</td>
<td>Transistor</td>
<td>Najm, etc.</td>
<td>• Sum of the Power {Dynamic (Transistor switching), Short circuit, Leakage current, \ldots}</td>
<td>SPICE PowerMill</td>
</tr>
<tr>
<td></td>
<td>Logic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Gate</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Circuit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RT</td>
<td>Architecture</td>
<td>Landman, Brooks, Bogliolo, etc.</td>
<td>• Sum of the Power {Adders, Multipliers, Controllers, Register files, SRAM's, \ldots}</td>
<td>Simple-Power Wattch</td>
</tr>
<tr>
<td></td>
<td>Architecture</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Behavior</td>
<td>Kumar, Raghunathan, etc.</td>
<td>• Sum of the Power {Processor, Bus, Memory, \ldots}</td>
<td>PowerPlay</td>
<td></td>
</tr>
<tr>
<td>System</td>
<td>Lidsky, etc.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SW</td>
<td>Instruction</td>
<td>Tiwari, etc.</td>
<td>• Sum of the Power {Instruction base cost, Inter-instruction cost, Other factors' (Cache miss, Pipeline stall, etc.) cost, \ldots}</td>
<td>JouleTrack Panalyzer</td>
</tr>
<tr>
<td></td>
<td>Function</td>
<td>Qu, etc.</td>
<td>• Sum of the Power {Built-in library functions, User-defined functions, Main functions, \ldots}</td>
<td></td>
</tr>
</tbody>
</table>
Model-based power estimation simulation framework

- SW Design Model (Specified with UML)
- HW Info
- Transform to Simulation Model
- Profiling

Simulation
- Search corresponding power consumption Fn.
- Accumulate calculated power consumption
- Calculate power consumption
- Power consumption estimation result

Function Library
- Power consumption function
- Parameter repository of power consumption Fn.

Currently simulated model element
- Find next model element
Applications of the power estimation technique

- Develop low power design patterns
- Identify and visualize the bottleneck of power consumption in an embedded software design model
Thank You .