
- Tutorial
2008. 9. 3

Ah-Rim Han

© KAIST SE LAB 2008
Contents

- Introduction
- Techniques and trends for embedded system design
- Power estimation and low-power design techniques for embedded software
  - Studies in various levels of abstraction
- Conclusion
- My research direction
Introduction (1/2)

- Embedded system
  - Is a special-purpose computer system designed to perform one or a few dedicated functions
Development of embedded system is getting difficult

- Shortening of the life cycle of the embedded products
- Increasing complex of embedded system design

We need the appropriate methodology and tool support for designing embedded systems!
Techniques and Trends for Embedded System Design
Conventional method

- System specification
  - Functional simulation

- Architecture decision by experts

- Prototyping
  - Test
  - Debugging, optimization

- High cost & inefficient design loop

→ Early separation of HW and SW
→ Long design time/ high cost/ hard to maintain and debug
New methods

- Platform-based design
- HW/SW codesign
- ...
Platform-based design (1/3)

- Referred first in the book\(^1\)

"an integration oriented design approach emphasizing systematic reuse, for developing complex products based upon platforms and compatible hardware and software virtual component, intended to reduce development risks, costs and time to market".

What is a platform?

- A library of components that can be assembled to generate a system at that level of abstraction
  - Hide unnecessary details and expose only relevant parameters for the next step

This layer wraps the essential parts of the software platform
- Programmable cores and the memory subsystem through an RTOS
- I/O subsystem through the device drivers
- Network connection through the network communication subsystem.
Platform-based design (3/3)

- Benefits
  - The design risks are reduced
  - The time to market is accelerated
  - The total cost would be lower
**HW/SW codesign (1/3)**

- Concurrent design of HW/SW components
- Evaluate the effect of a design decision at early stage by “virtual prototyping” to enable systematic design space exploration

※ Design Space Exploration (DSE) –
Find an optimal architecture for the given system functions to satisfy the requirements with the minimum overhead (performance, power, etc.)
HW/SW codesign flow

- System Specification
- Functional simulation
- Design Space Exploration (HW/SW Partitioning)
- Architecture Optimization
- Performance Estimation (Cosimulation/Co-verification)
- Interface Synthesis
- SW Synthesis
- Prototyping
- HW Synthesis
Benefits

- A significant performance improvement for embedded system design
  - Earlier architecture closure
  - Reducing risk by 80%
- HW/SW engineering groups to talk together
- Earlier HW/SW integration
- Reducing design cycle
  - Developing HW/SW in parallel
Embedded systems development has become mainly software-driven.

Automobile is an embedded system in which software plays an important role.

Therefore, systematic development techniques for Embedded Software (ES) are needed.
Critical aspects in ES development

- Design decisions at a higher abstraction level
  - Fast Design Space Exploration (DSE) in the early design steps

Validation/verification during the Software Development Life Cycle
Critical aspects in ES development (Cont’d)

- Automation under different models of computation
  - Model Driven Architecture (MDA)-based approach

Platforms: Web Services, ebXML, J2EE/EJB, CORBA, MS .Net, …
Energy Estimation and Low-Power Design Techniques for Embedded Software
Energy analysis in ES (1/3)

- Energy is one of the most important non-functional requirements for embedded system design.

- Energy cost of embedded system depends on
  - Not only hardware platform but also software.
State of the art

- Many energy measurement tools are available for lower level of the design at the circuit-level and the gate-level.
- However, little work has been done for analyzing power consumption from the point of view of software:
  - No concrete measure of the impact of their modeling decisions.
  - No support for software automation under different models of computation.
Energy analysis in ES (3/3)

- Energy studies in various levels of abstraction

<table>
<thead>
<tr>
<th>Level</th>
<th>Energy Estimation</th>
<th>Energy Optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction</td>
<td>V. Tiwari et al. (1994) “Power Analysis of Embedded Software: A First Step Towards Software Power Minimization”</td>
<td>-</td>
</tr>
<tr>
<td>Function</td>
<td>G. Qu et al. (2000) “Function-Level Power Estimation Methodology for Microprocessors”</td>
<td>-</td>
</tr>
</tbody>
</table>
Instruction-level energy estimation

- Basic unit of estimation – instructions
  - Fundamental level for S/W like gates for digital circuits
  - Practically applicable compared to circuit-/architecture-level power model
Tiwari’s work (2/3)

Energy estimation process

<table>
<thead>
<tr>
<th>Program</th>
<th>Base Cost ((m/A)) Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{Block } B1)</td>
<td></td>
</tr>
<tr>
<td>mov bp,sp</td>
<td>285.0 1</td>
</tr>
<tr>
<td>sub sp,4</td>
<td>309.0 1</td>
</tr>
<tr>
<td>mov dx,0</td>
<td>309.8 1</td>
</tr>
<tr>
<td>mov word ptr -4[bp],0</td>
<td>404.8 2</td>
</tr>
<tr>
<td>(\text{Block } B2)</td>
<td></td>
</tr>
<tr>
<td>mov si,word ptr -4[bp]</td>
<td>433.4 1</td>
</tr>
<tr>
<td>add si,si</td>
<td>309.0 1</td>
</tr>
<tr>
<td>mov bx,dx</td>
<td>285.0 1</td>
</tr>
<tr>
<td>mov cx,word ptr -4[si]</td>
<td>433.4 1</td>
</tr>
<tr>
<td>add bx,cx</td>
<td>309.0 1</td>
</tr>
<tr>
<td>mov si,word ptr -4[si]</td>
<td>433.4 1</td>
</tr>
<tr>
<td>add bx,si</td>
<td>309.0 1</td>
</tr>
<tr>
<td>mov dx,bx</td>
<td>285.0 1</td>
</tr>
<tr>
<td>mov di,word ptr -4[bp]</td>
<td>433.4 1</td>
</tr>
<tr>
<td>inc di,1</td>
<td>297.0 1</td>
</tr>
<tr>
<td>mov word ptr -4[bp],di</td>
<td>500.1 1</td>
</tr>
<tr>
<td>cmp di,4</td>
<td>313.1 1</td>
</tr>
<tr>
<td>jl L2</td>
<td>405.7(356.9) 3(1)</td>
</tr>
<tr>
<td>(\text{Block } B3)</td>
<td></td>
</tr>
<tr>
<td>mov word ptr -4[sum],dx</td>
<td>521.7 1</td>
</tr>
<tr>
<td>mov sp,bp</td>
<td>285.0 1</td>
</tr>
<tr>
<td>jmp main</td>
<td>405.8 3</td>
</tr>
</tbody>
</table>

Assembly/Machine Code -> Determination of Basic Blocks

Stall Analysis -> Basic Block Cost Estimation -> Global Program Cost Estimation

Execution Profiling -> Global Program Cost Estimation

Base Cost Table

Cache Penalty Estimation (Cache Simulation)
Overall instruction level power model

\[ E_p = \sum_i (B_i \times N_i) + \sum_{i,j} \left( O_{i,j} \times N_{i,j} \right) + \sum_k E_k \]

- \( B_i \): Instruction base cost
- \( O_{i,j} \): Inter-instruction cost
- \( E_i \): Other factors’ costs (cache miss, pipeline stall)
- \( N_i \) and \( N_{i,j} \): Number of execution
  - Determined via simulation since they depend on the execution path
Function-level energy estimation

- Basic unit of estimation – functions
- Build the power data bank which contains the power information of
  - Built-in library functions
  - User-defined functions
  - Main() function
  - Basic instructions
Energy estimation process

Input: a program with input data
Output: power estimate for the execution of the input program on a specific microprocessor core

- Execute only once for a fixed hardware to build "Power data bank"
- High efficiency without simulation
Tan’s work(1/2)

- Architecture-based energy optimization
  - Basic unit of transformation – components and connectors
    - Architecture style - OS-driven multi-process architectural style
    - Components - application processes, signal handlers, device drivers, etc.
    - Connectors - inter-process communication (IPC), synchronization mechanisms
Energy optimization process

- Evaluating the energy impact of atomic software architecture transformations through the use of energy macro-models
- Constructing sequences of atomic transformations that result in maximal energy reduction

Fei’s work (1/2)

- Program code-level optimization
  - Basic unit of transformation – block of source codes
  - Build control/data flow process network

Find-grained view - Functional call graph

Process level view - Processes, control/data flow and dynamic constructs (e.g., semaphores, IPC) are visible
Fei’s work (2/2)

- Energy optimization process

1. Initial program code
2. Extract control/data flow process network
3. Hierarchical process network
4. Apply transformation
5. Select best transformation

Initial process network (Energy is obtained using EMSIM\textsuperscript{[2]} which is an instruction-based simulator)

Transformations & constraints

Find possible transformations

Evaluate all transformations

OS energy macro-models

Optimized program code

Process P (f(\(f\))

Process Q (g(\(g\))

Process P’ (f’(\(f’\))

Original processes

Merged process
Research area is overlapping

- We (system and software engineers) are interested in same keywords
  - Embedded software, UML, design space exploration, high-level modeling, MDA, etc.

- Related conferences
  - DATE (Design, Automation and Test in Europe)
  - ISORC (International Symposium on Object/component/service-oriented Real-time distributed Computing)
  - MOMPES (International Workshop on Model-based Methodologies for Pervasive and Embedded Software)
  - …
Conclusion

- New methodology and tool support is required for developing embedded system
- Software plays an important role more than ever in an embedded system
- Keys to success in ES development
  - Design decisions in higher abstraction level
  - Automation under different models of computation
- Researches of analyzing power consumption from the point of view of software are getting much attention and studied
My future research direction would be...

- Model-based energy estimation and optimization
  - Strategies
    - UML modeling and high-level design space exploration
    - Software synthesis using an MDA-based approach
Energy modeling approach

- UML is for modeling embedded system
  - Application model (PIM) is constructed with Class Diagram, Interaction Overview diagram, and Sequence Diagrams
  - Platform model (PSM) is annotated with MARTE profile to specify real-time and QoS properties
Preliminary overview of the energy estimation approach

- **SW Modeling**
  - SW Design Model
  - UML to CFG Transformation
  - Simulation
    - EBU Detection
    - EBU Energy Accumulation
  - End of CFG
    - Estimated Energy Consumption

- **HW Modeling**
  - HW Design Model
  - VI and VSF Mapping & Energy Estimation
    - Mapping and Selected VI's Energy Estimation
    - Mapping and Selected VSF's Energy Estimation
  - VI Energy Value
  - VSF Energy Function
  - VI Energy Model

*EBU TYPE*
1. Action Semantic
2. Instruction
3. System Function

*Type Energy Modeling*
- "Action Semantic" Type Energy Modeling
- "Instruction" Type Energy Modeling
- "System Function" Type Energy Modeling
Model-based energy estimation (3/3)

- Application of the proposed energy estimation framework
  - Low-power design patterns
  - Identify and visualize
    - The spot of the energy bottleneck
    - The most efficient energy saving control-flow path
Thank You.
### Energy Behavioral Unit (EBU)

**Example**

- **Instruction type – synchronous message**

<table>
<thead>
<tr>
<th>SD</th>
<th>Control Flow Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>

- **EBU detection algorithm in CFG using OCL**

```java
boolean detectFunctionCall(CFG curNode) {
    if (curNode instanceof MessageOccurrenceSpecification) {
        Message m = curNode.message;
        if (m.messageSort == synchCall) {
            if (m.sendEvent.covered == m.receiveEvent.executionFinish.message.receiveEvent.covered && m.receiveEvent.executionFinish.message.messageSort == reply)
                return true;
        }
    }
    return false;
}
```

→ **EBU in SD and CFG**