Introduction to the Design of Embedded System

2008. 12. 3
Presented by Sang-Uk Jeon

ⓒ KAIST SE LAB 2008
Contents

- Part I – What is Embedded System?
- Part II – Designing Embedded System
- Part III – Software Engineering in Embedded System Design

© KAIST SE LAB 2008
Part I. What is Embedded System?
Part I. What is Embedded System?

Definition of Embedded System*

- A combination of computer hardware and software, and perhaps additional mechanical or other parts
- Designed to perform a dedicated function
- In some cases, part of a large system or product
  - e.g.) Antilock Braking System (ABS) in a car

Part I. What is Embedded System?

Characteristics of Embedded System*

- Single-functioned
  - Executes a single program, repeatedly
- Tightly-constrained
  - Low power, small, fast, etc.
- Reactive and real-time
  - Continually reacts to changes in the system’s environment
  - Must compute certain results in real-time without delay

Part I. What is Embedded System?

Categorization of Embedded System

- Computation oriented
  - MP3 player, MPEG decoder, etc

- Control oriented
  - Home appliances, industrial controller, safety-critical controller

- Hybrid (computation + control)
  - Portable information devices
    - e.g.) Cellular phone
  - Networked multimedia applications
Part II. Designing Embedded System
Issues in Embedded System Design

- Top-priority design goal
  - Construct the system with desired functionality
- Design issue
  - Simultaneously optimize numerous design constraints
    - Size, performance, power, flexibility, etc.
Part II. Designing Embedded System – Traditional Embedded System Design

Traditional Embedded System Design

- System Specification
  - Designers partition the system into hardware and software early in the flow

- HW & SW Partition
  - HW and SW engineers design their respective components in isolation
  - HW and SW engineers do not talk to each other

- HW Development

- SW Development

- SW Development

- Prototype Test
  - Integration problems
  - High cost and long iteration

- Prototype Test passed
  - Need new methodology!!
    - HW & SW co-design
    - Platform-based design
    - ...

© KAIST SE LAB 2008
Part II. Designing Embedded System – HW & SW Co-Design

HW & SW Co-Design

- System Specification
- HW & SW Partition
- Design Space Exploration
  - HW Model
  - SW Model
  - Map HW & SW
    - Evaluation
      - Evaluation subject: Performance, energy, etc.
      - HW Synthesis
      - SW Synthesis
      - HW & SW Integration

* Synonym * Architecture Platform
* Synonym * Application Functionality

ⓒ KAIST SE LAB 2008
Design Space Exploration (DSE)*

- Finding the optimal design of software and hardware
  - That satisfies given design objectives

**Problem Space**
*Characteristics of SW & HW*
- Software functionality
- Hardware parameters
  - Processor architecture
    - Clock rate
    - Cache size
    - ...
  - Memory architecture
    - Page replacement policy
    - ...

**Solution Space**
*Design objectives*
- Performance objective
- Energy objective
- ...

---

Decides whether each functionality is implemented in HW or SW

Requirements: Summing up 100 values

**Alternative 1**
- Software implementation
  - ADD v1, v2
  - ADD v1, v3
  - ...
  - ADD v1, v100
- Hardware support
  - More flexible software

**Alternative 2**
- Software implementation
  - ADD v1, v100
- Hardware support
  - Smaller software size
  - Better performance
Software Model

- Kahn process networks
- Directed Acyclic Graphs (DAG)
- DAGs with periods and deadlines
- Synchronous data flow
- Control data flow graphs and dynamic data flow
- High-level programming language
- Co-Design Finite State Machines
- Communication analysis graphs
- Click model of computation
- Transaction Level Modeling
  - SystemC
- Hierarchical and heterogeneous models of computation
  - Ptolemy framework, Metropolis meta-model language, etc.
Hardware Model

- Abstract models
  - Instruction-accurate model
  - Task-accurate model
  - Non-linear, accumulative service descriptions

- Executable models
  - Micro-architecture templates
  - Hardware description language
  - Architecture description language
Mapping Software and Hardware

- Binding software tasks to hardware building blocks
  - May require additional jobs such as
    - Rewriting/adapting software code
      - To link required interface of software and provided interface of hardware
    - Compilation of the software onto the hardware

Software Tasks

Hardware blocks

© KAIST SE LAB 2008
Evaluation (1/3)

- Simulation-based methods
  - Cycle-accurate simulation
    - Cycle 1 (do Inst1)
    - Cycle 2 (do Inst2)
    - Cycle 3 (do Inst2)
    - Read Memory
      - Assume that it consumes 3 cycle
  - System-level simulation
    - Read Memory
      - Done! (3 cycle)
Analytical methods

- Static profiling
  - Complexity analysis of algorithms, the dependency analysis of function call graph, etc.

- High-level synthesis
  - Find an optimal mapping for the software tasks to the hardware
    - Exact method such as Integer/mixed linear program formulations, heuristics, evolutionary algorithms, etc.
Combination of simulation-based and analytical methods

- **Trace-based performance analysis**
  - A trace contains all memory access
  - Initial simulation
  - Use the trace to calculate performance

- **Analytical models with initial, calibrating simulation**
  - Simulation results
  - Cache miss rate: 10~20%
  - Obtain ranges of performance factor
  - Exhaustive simulation
  - Feed the information into the analytical model

\[
P = \frac{A}{B} \times \ldots
\]

\[
Q = C \times 100 + \ldots
\]

\[
\text{Performance factor } = P + Q
\]
Frameworks for DSE

- **System-level**
  - Metropolis
  - Mescal
  - StepNP
  - SPADE
  - Artemis
  - MILAN
  - MESH
  - SEAS
  - Incisive-SPW
  - CoCentric System Studio

- **Micro-architecture centric**
  - Mescal/Tipi
  - ASIP-Meister / PEAS-III
  - EXPRESSION
  - LisaTek
  - Chess / Checkers
  - MaxCore & MaxSim
Example – Metropolis (1/4)

**Functionality modeling**

---

<table>
<thead>
<tr>
<th>Process Src</th>
<th>Medium S</th>
<th>Process Sink</th>
</tr>
</thead>
<tbody>
<tr>
<td>port Out out; void thread() { out.send(data); } void doSomething() {}</td>
<td>void send() { ... } void receive() { ... }</td>
<td>port In in; void thread() { in.receive(data); }</td>
</tr>
</tbody>
</table>

Port Out { void send() {} }

Port In { void receive() {} }

© KAIST SE LAB 2008
Example – Metropolis (2/4)

- Architecture modeling

```cpp
Process Task {
  port TaskToCPU taskToCPU;
  void thread() {
    void read() {
      ...
    }
    void write() {
      ...
    }
    void execute(int n) {
      // Consume n CPU cycles
    }
  }
}
```

```cpp
Medium CPU {
  void read() {
    // read from memory
  } void write() {
    // write to memory
  }
}
```
Example – Metropolis (3/4)

Mapping

Mapper {
    ...
    constraint {
        ...
        Src.send => Task1.write;
        Src.doSomething => Task1.execute(50);
        Sink.receive => Task2.read;
        ...
    }
    ...
}
Example – Metropolis (4/4)

- A part of the output in “Simple case study” example

```
SystemC 2.1.0v1 —- Mar 25 2008 22:54:09
Copyright (c) 1996–2005 by all Contributors
ALL RIGHTS RESERVED
SwTask1 running Cpu write cwp1 beg
SwTask0 running Cpu write cwp1 beg
CpuScheduler1 ### Process:– SwTask1 Address :– 0x8f49b50 Event:– 0x9015120
CpuScheduler0 ### Process:– SwTask0 Address :– 0x8f474f8 Event:– 0x90151e8
CpuScheduler0 Resolve: FindWinner SwTask0
CpuScheduler1 Resolve: FindWinner SwTask1

In CpuScheduler0, make request to GTime, e: 0x90155a0, Time: 5e-09

BusScheduler0 ### Process:– SwTask1 Address :– 0x8f49b50 Event:– 0x90156d8
BusScheduler0 Resolve: FindWinner SwTask1
In BusScheduler0, Current Global Time is: 5e-09
In BusScheduler0, make request to GTime, e: 0x90156d8, Time: 5e-09
```
Main idea

- Reusing & facilitating a common design to a variety of different applications
**What is the Platform?**

**Platform**
- A library of components
  - To generate a design at certain level of abstraction

**Platform instance**
- A set of components
  - Selected from the library(platform)
  - Parameters are set


© KAIST SE LAB 2008
What is the Platform? (2/2)

- Multiple abstraction levels of the platform
  - Each platform instance in the abstraction levels can be reused

![Diagram showing multiple abstraction levels of the platform with processor, memory, and bus components, along with various platform instances and interfaces like ARM, MIPS, DDR SDRAM, ISA, DDR SDRAM, PCI, DDR-200, DDR-333, DDR2-533, DDR2-800, 8-bit ISA, 16-bit ISA, PCI 2.2, and PCI 3.0.](image-url)
Part II. Designing Embedded System – Platform-Based Design

Design Process

- Meet-in-the-middle process
  - Combination of top-down and bottom-up approach
A Platform for Software Reuse

API Platform*

Part III. Software Engineering in Embedded System Design
Challenge in Embedded System Design

- Consideration of flexible implementation
  - To address rapidly changing & increasing requirements
  - Sacrificing some degree of performance

Increasing usage of programmable elements instead of ASICs (Application Specific Integrated Circuits)

Increasing complexity of application (software)
Research Issues (1/3)

- In the view of the software process*

  - Issue 1: Coordination of embedded software development process with other sub-processes
  - Issue 2: Specialized software development process for dealing with non-functional requirements (Memory, power, real-time requirements, etc.)

Part III. Software Engineering in Embedded System Design

Research Issues (2/3)

- In the view of the software model

  - HW & SW Partition

  - HW Model (Abstraction Lv. 1)
  - SW Model (Abstraction Lv. 1)

  - Map HW & SW

  - Evaluation

  - HW Model (Abstraction Lv. n)
  - SW Model (Abstraction Lv. n)

  - Map HW & SW

  - Issue 1: DSE in a more abstract level
  - Issue 2: Performance/energy-aware refactoring
  - Issue 3: Performance/energy-related metric
  - Issue 4: Model-Driven Development

© KAIST SE LAB 2008
Research Issues (3/3)

In the view of the software code

- HW & SW Partition
- HW Model
- SW Model
- Map HW & SW
- Usually SystemC code
- Evaluation failed
- Evaluation*

Issue 1: UML to SystemC

Issue 2: Application of code-level software engineering techniques
- Testing
- Clone detection
- Reverse engineering
- Metric
- ...

Part III. Software Engineering in Embedded System Design
References


Discussion

Q & A