RTOS and Codesign Toolkit for Multiprocessor System-on-Chip

Shinya Honda, Hiroyuki Tomiyama, Hiroaki Takada

Graduate School of Information Science, Nagoya University
honda@ertl.jp
Outline

- Motivation
- Real-Time Operating Systems for Multiprocessor systems
  - FDMP kernel
- Codesign Toolkit for MPSoCs
  - Systembuilder
- A Case Study
- Current Status
- Summary
Motivation

- Multiprocessor design have become popular in embedded domains. Increasing the number of processors is generally more power/performance efficient than increasing the clock frequency.

- RTOS have become commodity tools in embedded systems. Over 80% of embedded system development project in Japan have used RTOS.

*In the development of embedded systems running on MPSoCs, RTOS’s for MPSoCs are necessary.*

- RTOS specification and implementation for MPSoCs: FDMP kernel. Support asymmetric MP systems (AMP).
Motivation (cont.)

- Mapping application tasks onto processors (design space exploration)
  - One of the most important decision in the design of MPSoCs

  *In order to achieve the best mapping, designer has to accurate estimation of design quality for each candidate mapping*

- Design Space become large
  - With hardware/software codesign and C-based behavioral synthesis
  - Mapping to hardware modules is possible

- System-level design toolkit: SystemBuilder
  - Evaluate the quality of mapping very quickly
  - Start with system specification in the C language
  - Takes the system specification and mapping directive as input and Generates implementation
Real-Time Operating Systems for Multiprocessor systems
Classification of Multiprocessor Systems

From a view point of RTOSs, multiprocessor systems are classified several type

- Tightly-coupled MP system with shared memory
  - Classified into two types
    - Symmetric multiprocessor (SMP)
    - Functionally distributed multiprocessor (FDMP)
    - Asymmetric MP systems (AMP)
  - Using in Embedded System.

- Loosely-coupled MP system (Distributed MP systems)
  - latency of inter-process communication is very slow
  - Using application level inter-process communication which support large unit size
Symmetric multiprocessor (SMP)

- Every processor can access all resources in the system: memory, peripherals, etc.
- An application task can be executed on any processor
- An example of SMP Hardware: ARM MPCore

Up to 4 processors
- Each processor can access all resources

*From ARM WebSite*
Functionally distributed multiprocessor (FDMP)

- A processor can access only a limited set of resources in the system
- Each processor assigned specific role (faction) in many case
- An application task is statically allocated to a specific processor
- An example of FDMP Hardware: Toshiba MeP

MPEG2 Codec LSI
- Each processor has specific hardware (Co-processor).

* From Toshiba WebSite ”Introduction to MeP(Media Embedded Processor)”
Multiprocessor Systems for Embedded Systems

Many embedded systems are dedicated to specific applications.

Application tasks can be statically mapped onto processors in such a way:

- Processor loads are balanced well.
- Inter-processor communication is minimized.

*FDMP architecture has advantage in many embedded systems*

- Cost, Power consumption, Realtime performance.
- Developed RTOS for this architecture.
uITRON Specification

RTOS specification for FDMP systems is based on uITRON Specification.

A standardized specification of RTOS kernel for small- to mid-scale embedded systems.

- Defines a set of API functions (service calls)
  - The fundamental subset is called Standard Profile (81 API)
  - ITRON is just specification, no software product.

Current status

- Most popular RTOS specification in Japan
  - 20 - 30% of embedded systems use uITRON
  - Especially in consumer electronics.
- The latest release is uITRON version 4.0.
FDMP Kernel has been implemented based on TOPPERS/JSP Kernel

A reference implementation of the Standard Profile of uITRON 4.0

- Initially, developed by our laboratory
- Currently, maintained by Non-Profit-Making Organization TOPPERS Project
  - Incorporated in September 2003. (President: Prof. Takada)
  - http://www.toppers.jp/
  - >100 members (universities, companies, and individual volunteers)
- Supported processors include
  - SH1/3/4, H8, M32R, ARM7/9, MIPS3, NiosII, V850, etc.
TOPPERS/JSP Kernel (cont.)

- Compact and highly portable
- Free, open source software
  - Can be used for research, education, and commercial purposes.
- Production quality
  - Actually, used in a number of products

Karaoke microphone (Panasonic)  Audio Interface (Roland)  Ink-jet printer (Brother Industries, Ltd)
Design Principles of FDMP Kernel

- Problems for Application level inter-processor synchronization and communication
  - Programming such programs are difficult and cause of time-consuming and error-prone
  - Application programs need to be modified every time mapping is changed

- APIs
  
  *Same APIs for both intra-processor synchronization and communication and inter-processor ones*

  - Inter-processor synchronization and communication can be realized with same uITRON Specification API

- Other requirements
  
  - Predictability
  - Scalability for number of processor
Classification of Objects

Each kernel object (tasks, semaphore, etc.) belongs to one of the processors

- **Static mapping**
  - Allocation of kernel objects to processor is statically
  - Tasks and Interrupt handlers can be executed only the mapped processors
System Status

- CPU locked state
  - System state in uITRON Specification
  - Exclusively executing a specific task
  - Interrupt handlers are not started
  - Dispatching does not occur
  - Often used for Mutual exclusion

- CPU locked state in FDMP Kernel
  - Controlled processor-by-processor independently
  - Scalability for number of processor
  - Mutual execution should be implemented using synchronization object
    - semaphores, eventflag, etc.
FDMP Kernel Implementation

Lock unit
- Mutual exclusion unit for kernel object control blocks (TCB, etc.) in multiprocessor environment
- Design two lock unit (a Task Lock and an Object lock) for each processor
  - Minimize dead-lock avoidance system call
  - Scalability for number of processor

Support Target
- ARM MPCore, Altera NiosII, Xilinx Microblaze, Toshiba MeP
Evaluation

Evaluate code size and performances of FDMP Kernel

Target system
- Altera NiosII/s 50MHz,
  - Soft-core processor for FPGA
  - Each processor has local memory

Avalon bus
- Star-type network
  - No contention happens as long as the processors access their local memory
Evaluation : Code size

Comparison of code size between JSP Kernel and FDMP Kernel

- Not include application code and data for kernel objects
- Text section
  - 60% larger than JSP Kernel
  - A routine for acquiring and releasing lock is inserted in all APIs
  - A new routines for avoiding deadlocks is added in some APIs
- Data and Bss section
  - An increase is trivial
  - TCB(Task Control Blocks) is extended by 6bytes

<table>
<thead>
<tr>
<th>Kernel</th>
<th>text</th>
<th>data</th>
<th>bss</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSP</td>
<td>26671bytes</td>
<td>5bytes</td>
<td>68bytes</td>
</tr>
<tr>
<td>FDMP</td>
<td>42707bytes</td>
<td>6bytes</td>
<td>76bytes</td>
</tr>
</tbody>
</table>
Evaluation: Performance

Evaluate execution times of two system calls

- **System calls**
  - `wup_tsk`: Wakes up task in the wait state
  - Acquires a one lock: a task lock
  - `sig_sem`: Released for semaphore
  - Acquires two locks: a task lock and an object lock

- **Conditions**
  - With invoked task dispatch
  - Without invoked task dispatch
Evaluation: Performance (cont.)

- Execution time becomes longer even in case of intra-processor system call
  - Additional routines for mutual exclusion
  - Data structures being more complicated
- In case of system calls with dispatch, FDMP (inter-processor) are longer than those of FDMP (intra-processor)
- The increased overhead for dispatching a task on different processor

<table>
<thead>
<tr>
<th></th>
<th>Without Task Dispatch</th>
<th>With Task Dispatch</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Kernel</strong></td>
<td><strong>wup_tsk</strong></td>
<td><strong>sig_sem</strong></td>
</tr>
<tr>
<td>JSP</td>
<td>5usec</td>
<td>7usec</td>
</tr>
<tr>
<td>FDMP (Intra-processor)</td>
<td>9usec</td>
<td>11usec</td>
</tr>
<tr>
<td>FDMP (Inter-processor)</td>
<td>9usec</td>
<td>17usec</td>
</tr>
<tr>
<td></td>
<td>10usec</td>
<td>18usec</td>
</tr>
</tbody>
</table>
Codesign Toolkit for MPSoCs
**SystemBuilder**

SystemBuilder is a system-level design environment for MPSoCs

**Main Feature**

- System-level description in C language
- SW/HW mapping by human designers
- Generation of software running on RTOS
- Generation of synthesizable hardware with a commercial tool
- Automatic SW/HW interface including interface circuits and device driver

*Both single processor and multi processor supported*

- SW/RTOS/HW cosimulation at various abstraction levels (single processor only)
- FPGA implementation
SystemBuilder (cont.)

Design Flow

- Application description with C language
- Behavioral verification of the application description
- Implementation synthesized with designated mapping (architecture)
- Performance estimation of the implementation
- Repeat the process if the performance is insufficient
System Description in SystemBuilder

An application is described as a set of sequential processes and communication channels among them.

Processes (P)
- Unit of concurrent execution
- Unit of mapping
- Written in the C language
- SW : task
- HW : module with single FSM

Communication Primitives (CP)
- Three kinds of fundamental channel
- Non-Blocking (Register)
- Blocking (FIFO)
- Memory

An example of system
System DeFinition (SDF) file

- The overall structure of application specification
- Mapping of processes to processing elements (processors and hardware module)

SYS_NAME = test

<table>
<thead>
<tr>
<th>SW(CPU1) = P1, P4</th>
<th>SW(CPU2) = P2</th>
<th>HW = P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCPRIM CP1, SIZE = 32, DEPTH = 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCPRIM CP2, SIZE = 32, DEPTH = 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NBCPRIM CP3, SIZE = 32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NBCPRIM CP4, SIZE = 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MEMPRIM CP5, SIZE = 8, DEPTH = 20</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

BEGIN_PROCESS
NAME = P1
FILE = “p1.c"
USE_CP = CP1(OUT), CP3(INOUT), CP5(OUT)
END

BEGIN_PROCESS
NAME = P2
FILE = “P2.c"
USE_CP = CP1(IN), CP4(OUT), CP5(IN)
END

....
Synthesis

SystemBuilder takes an SDF file and C programs as input, and automatically generates implementation.

Software Synthesis

- Process (P1,P3,P4)
  - Translated to software tasks for FDMP Kernel
- CP(CP1,CP3)
  - Replaced with corresponding synchronization/communication services calls of the FDMP Kernel
Synthesis : Hardware Synthesis

Hardware Synthesis

- Process (P3)
  - Synthesized with a commercial behavioral synthesis tool (eXCite)
  - SystemBuilder automatically executes eXCite to generate RTL

- CP
  - Translated into HW/HW communication circuit
  - registers, FIFO, memory
**Synthesis : Interface Synthesis**

**Interface Synthesis**

- **CP(CP2, CP4, CP5)**
  - **Software side**
    - Device drivers are generated
    - Read/Write interface
    - Interrupt Handler
  - **Hardware side**
    - HDL description of interface circuit are generated
    - Register, FIFO, Memory
Synthesis: Interface Synthesis (cont.)

- HW/SW communication is based on memory mapped I/O accesses
  - Automatic address assignment to the generated storage
  - Address decoder circuit and an interrupt controller are synthesized
- These synthesis steps are completely automated

Can explore a large number of different mappings in short time
A Case Study
Case study on design space exploration

Evaluate the effectiveness of SystemBuilder

- JPEG decoder application
  - Consists of seven processes
  - Four processes, i.e., iquantize, idct, pshift, and yuv2rgb, can be implemented in software or hardware

- Target
  - Single-processor and dual-processor architecture platforms
  - Xilinx Virtex-2, Microblaze
Case study on design space exploration (cont.)

- Synthesized and Evaluated 12 designs
  - With different mapping and both single and dual processor platform
  - Const-performance trade-offs

*Evaluation can be done only a day by single designer*
Current Status
Current Status on SystemBuilder

- Extending Systembuilder
  - Abstract (HW/SW) interfaces description with higher abstraction and synthesis technique
  - Architecture explorations

Interface description with higher abstraction

- goal
  - Interface description in the system level should be independent of architecture

- Current version of CPs do not achieve this goal, because abstraction level of interface description is still low
  - We are defining communication primitives and APIs with higher abstraction.
Current Status on SystemBuilder (cont.)

Architecture explorations

- Support wider design space (multi-bus, multi PEs)
- Takes as input not only an application description but also an architecture template
Summary
Summary

- Principles and techniques for design and implementation of RTOS for embedded multi processor (FDMP Kernel)

- System-level design toolkit for rapid design and evaluation of embedded multi processor (SystemBuilder)

Current Status

- Abstract (HW/SW) interfaces description with higher abstraction and interface synthesis from it
- Wider space for architecture explorations

FDMP Kernel is available as a open source software from
http://www.toppers.jp/
Thank you for listening!