A Flexible Virtual Development Environment for Embedded Systems

2007. 6. 21.

Sang-Young Cho, Yoojin Chung, and Jung-Bae Lee*

CSE, Hankuk University of Foreign Studies
*CI, Sunmoon University
Contents

- Introduction
  - Why and What is Virtual Development Environment
  - Examples and Our Approach

- Related Research
  - ARMulator and SystemC Environment

- Design and Implementation
  - Extension of ARMulator Environment
  - Extension with SystemC Models
  - Implemented VDE and uC/OS-II Porting
  - Verifying VDE

- Conclusions
Introduction

Why Virtual Development Environment?

- Time-to-market is very crucial for embedded systems
- Most embedded systems contain hardware IPs and software IPs
- Traditional development flow
  - Application software design is not started until the IPs are integrated
  - Development cycle is too long
- Virtual prototype approach
  - Virtually built inside a computer, and simulates real hardware
  - Software development can be performed without tangible hardware
  - Shorten the development time
    - Initial verification of SW/HW
    - Predicts performance values and guides a final design

Virtual Development Environment (VDE)

- Virtual hardware model & Simulation engine
- Software development tools & Software models
Introduction

Virtual Development Environment Examples

Virtual Platform
- Commercial Configurable VDE of Virtio
- Various cores of ARM, X-Scale, MIPS
- Software design, development, and verification

MaxSim
- Commercial VDE of ARM for SoC development
- Support SystemC
- ESL (Electronic System Level) Tool (SW+HW development)

Visual ESC
- Commercial VDE of Summit
- Processor models for ARM, MIPS
- ESL tool

- Expensive
- Tightly integrating hardware simulation & software development tools
- Limited flexibility of using hardware model and software tools
Introduction

Our Approach for Virtual Development Environment

- Useful and Cheap Solution
  - For ARM processor cores \( \leftarrow \) over 70% market-share
  - ARMulator based VDE \( \leftarrow \) ADS 1.2
    - Support upto ARM10 and Xscale
  - Hardware IPs for PDA
  - uCOS-II based programming

- Flexible Environment
  - SystemC Engine is attached to ASB bus
    - SystemC HW IP models
  - SystemC Engine is attached to AxD with RDI 1.5.1
    - Only SystemC models
  - User Interface for LCD panel, UART, LED display

Describe a VDE implementation for SW development based on ARMulator and SystemC
Related Studies: ARMulator Environment

- ARM’s virtual software development environment
  - Cycle-based instruction set simulator
  - Basic memory model
  - Can be extended
In usual ARMulator environment
- AxD of ADS1.2 or RealView Debugger of RVDS 3.0
- Processor cores + Basic hardware IP’s
- Profiler, MMU, Semihosting
Related Studies: SystemC

SystemC?

- C++ class library to support system level design
- SystemC ver. 2.x: register transfer, algorithm/function level
- Coming version: will support real-time OS and analog circuit

SystemC Design Methodology

SoC development without SystemC

SoC development with SystemC

SystemC Model → Simulation → Improvement → Synthesis → Remained works
SystemC Structure

System Modeling with SystemC

- Consists of modules and processes with hierarchical structure
- Module includes other modules or processes (Container class)
- Processes model functionalities and defined within a module
- Port: Module has ports and modules are connected via ports
- Signals connect modules through ports
- Clock: SystemC’s special signal for a system clock
- Cycle-based simulation: untimed model with clock cycle accuracy

SystemC’s System Modeling
Design and Implementation

Extension of ARMulator Environment

- Refer to S3C2410
  - for PDA
- ARMulator
  - ARM920T
    - MMU, Cache
  - ASB, APB
    - SystemC
  - MC, DMA, UART
    - TIMER, WDT, GPIO, Bridge
- SystemC
  - LCDC, INTC

Overall Structure
Design and Implementation

SystemC Extension

Implemented Module using SystemC

SystemC Module

SystemC Module

SystemC Module

Simulation engine

Processor Core

Flat Memory

Armul_bus

Csimul Class

Main Module

① Initialize

② SystemC Init

③ init feedback

④ init feedback

⑤ bus operation

⑥ operation

⑦ states feedback

⑧ states feedback

signals

signals

signals
Design and Implementation

SystemC Extension

- In InitializeModule() function of Armul_bus
  - SystemC modules are initialized by Csimul class
  - SystemC.lib is modified
    - Main() → sc_main()
    - Clock is synchronized with ASB clock
- Csimul behavior
  - Generates modules
  - Make sc_signal to control input/output wires of modules
  - Connect signals after a main module in SystemC is made
  - Create functions for read/write of connected modules
  - During simulation, a callback function is called by Armul_bus
  - Allow simulation result to be reported to Armul_bus

SystemC engine is connected to ASB bus
Design and Implementation

Peripheral Features of the Implemented Environment

- ARMulator
  - Timer2
  - WDT2
  - DMAC2
  - UART
  - GPIO
  - MC
  - Bridge
  - SystemC

- SystemC
  - LCDC
  - NTC

- Windows
  - LCD panel
  - LED display
  - UART interface
  - PWM log
Design and Implementation

Porting uC/OS-II

uC/OS-II based Application

Application Dependent: OS_CFG.H, INCLUDE.H


Processor Dependent: OS_CPU.H, OS_CPU_A.S, OS_CPU_C.C

uC/OS-II

ARMulator

Processor Model (ARM7,9,10,11 + Cache)
Configurable Memory Model
Decoder

Tracer Profiler
MMU
Semihost

Timer
Interrupt Controller
Time tick
Watchdog Timer

Timer2
Watchdog Timer2
DMAC2

UART
SystemC Engine

GPIO Port
Memory Controller
Bridge

Window Program

UART Interface
LED Display
PWM Log

SystemC Modules

LCD Controller
Interrupt Controller

Window Program

LCD Panel
### Design and Implementation

#### Testing the implemented VDE

<table>
<thead>
<tr>
<th>Testing Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS: Microsoft Windows XP</td>
</tr>
<tr>
<td>Compiler: Microsoft Visual C++</td>
</tr>
<tr>
<td>Debug Controller: AXD Debugger of ARM Developer Suite v1.2</td>
</tr>
<tr>
<td>Test sample program: CodeWarrior of ARM Developer Suite v1.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3-Task Test Program</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main() creates TASK1 → TAS1 creates TASK2, TASK3</td>
</tr>
<tr>
<td>TASKs are moving side-to-side with different delay values</td>
</tr>
<tr>
<td>Each TASK draws its image to Image Buffer</td>
</tr>
<tr>
<td>Can verify scheduling with timer and interrupt controller</td>
</tr>
<tr>
<td>Can verify LCD displaying with ASB bus, LCD controller and LCD panel</td>
</tr>
</tbody>
</table>

With small sample programs
- Verify GPIO, DMAC, UART
Design and Implementation

Testing the implemented VDE
Conclusion

- Virtual Development Environment (VDE)
  - Provide embedded software development environment without real hardware → Reduce embedded system development cost

- We implemented a flexible VDE with ARMulator and SystemC models
  - Target processor core → adapts ARM920T processor core widely used in commercial
  - Debugger → ARM’s AxD
  - Extension of ARMulator: TIMER, WDT, MC, DMAC, UART, GPIO, SystemC engine
  - SystemC Module → LCD Controller, Interrupt Controller
  - User Interface → LCD panel, LED display, UART int., PWM logging
  - uC/OS-II Porting → Multi-threaded application

- Benefits of the implemented VDE
  - Multi-modeling
    - ARMulator model and SystemC model
  - Multi-threaded programming
    - With uC/OS-II API
  - Construct cost is very low
    - ADS 1.2 with public SystemC models