Code Generation for QEMU/SystemC
Cosimulation from SysML

MeCoES Workshop – Oct. 7th, 2012

Fabian Mischkalla, Da He, Wolfgang Mueller
University of Paderborn/C-LAB, Germany
Introduction

- SysML becomes important for early specification and analysis
  - Component-oriented design
  - Separated views (e.g. Structure/Behavior)
  - Functional deployment (e.g. SW→HW)

- Executable models for validation and synthesis
  - Virtual Prototyping (QEMU/SystemC)
  - FPGA-based Prototyping (Xilinx)

⇒ A SystemC code generation framework (in Artisan Studio®)
Agenda

- Introduction
- SysML-based Modeling
- Multiple code generation
  - Hardware components
  - Platform design
  - Software integration
- QEMU/SystemC cosimulation
- Summary
Component-based SysML Modeling

- Based on structural SysML + domain-specific extensions
  - Codegeneration (Simulation entry, Top-Level, …)
  - Configuration (AddressSpaces, MemoryMaps, OS, …)
Component Interface Modeling

- Modular interfaces for synthesizable protocol specifications
  - Pin-level encapsulation
  - Modularity by method calls

- SysML representation
  - Blocks/part references/flowports
  - Standard operations for detailed protocol implementation

- Application by SysML allocation technique
  - Allocated ports and connectors
Code Generation Overview

- A M2T transformation scheme
  - **Input**: User Model, Generator Model
  - **Output**: Executable Environment

- Generation rules are modeled in
  - Template language (SDL)
  - UML classes
  - → C++ code → DLL

- “Transformation” process in two steps

- Complete automation of cosimulation
System Platform Generation Support

SysML model
Software Integration Support

Introduction
SysML Modeling
Code Generation
Cosimulation

Software Integration Support

<<Component>>
SWComponent

Driver function (Generated)
• User-space device access
• Applies mmap syscall
• Read/write API

Code (referenced)
• UML artifacts
• File paths

Memory map (Generated)
• Address range
• Offset of each I/O register

Makefile (Generated)
• Cross-compilation to application binaries

QEMU

ISS
Target SW + OS

Script (Generated)
• Generates file system structure
• Integrates application binaries
• Configures QEMU with filesystem and defined OS images
Cosimulation Generation Support

Introduction
SysML Modeling
Code Generation
Cosimulation

QEMU Plugin (Generated)
- Creates Inter-Process Communication (IPC) channels
- Registers user-defined devices via QEMU API
- Implements IO callbacks (byte, word, long)

Makefile (Generated)
- Creates plugin Dll loadable at QEMU runtime

<<processor>>
cpu_0:CPU

Platform Plugin
ISS

QEMU
QEMU/SystemC Cosimulation Architecture

- QEMU and SystemC are separated processes
- Data transfer via shared memory
  - IO operations (read/write)
  - Interrupts
- Cosimulation infrastructure supports data transactions for speed-up
  - Command (read/write)
  - Address
  - Data lengths (in bytes)
QEMU/SystemC Synchronization

- SW/HW interaction by blocking interface
  - QEMU executes asynchronous
  - Handshaking protocol (i.e. status register polling)
  - HW interrupts

- Synchronization between SystemC and OS processes
  - SystemC yields periodically (i.e. wait(time))
  - Asynchronous update phase request from OS thread
Example Scenarios (Image processing domain)

- **Smart camera system** for Automatic License Plate Recognition (ALPR)
  - Originally by Thales Security Solutions & Services
  - Reengineering of architectural model (up to ~70% reduction by modeling reuse)

- **Embedded Video Detection** (by GreenSocs Ltd.)
  - Image acquisition via SW
  - Preprocessing and object labeling pipeline
  - Total code generation gain (~60%) (depends on ratio of behavioral code)
Summary

- Using SysML for specification and configuration of SystemC-based execution models

- The framework covers …
  - SysML modeling rules for component-based designs
    - Hardware components (synthesizable SystemC)
    - Software components and artifacts
    - Processor components
  - Code generation and configuration for executable models
  - QEMU/SystemC based cosimulation environment

Thank you!