A Flexible SystemC Simulator for Multiprocessor Systems-on-Chip

Luca Benini
Davide Bertozzi
Francesco Menichelli
Mauro Olivieri

DEIS - Università di Bologna
DEIS - Università di Bologna
DIE - Università di Roma “La Sapienza”
DIE - Università di Roma “La Sapienza”
Summary

• Introduction
  – Design concept
  – Overview of the AMBA bus

• System architecture
  – Processing unit
  – AMBA AHB bus and interfaces

• Simulation
  – Bus contention and test applications
  – Signal tracing

• Conclusions and future works
Project description:
A SystemC shared memory multiprocessor simulator composed by multiple instruction level models of cached ARM cores connected to a bus supporting multiple master arbitration.
Introduction

Design basics
- SystemC 1.0.2 library
- SWARM simulator (C++ class):  
  http://www.dcs.gla.ac.uk/~michael/phd/swarm.html
- Multiple master bus (AMBA) SystemC implementation
- RedHat Linux 7.2 and gcc 2.95.3
- UClinux single processor 2.4.0 kernel
- gcc 3.0.4 ARM cross-compiler

Why SystemC based design?
- re-use of C/C++ code
- modularity
- standard interface between modules (SystemC)
Outlines of the AMBA bus

AMBA (Advanced Microcontroller Bus Architecture) is an on-chip communication standard for high-performance embedded microcontroller supporting multiple units able to initiate read and write operations (bus masters)

- AHB: high performance, high clock frequency system modules
- APB: low power peripherals

Figure: AHB and APB in a typical AMBA system
Full SystemC AMBA AHB bus implementation
- cycle accurate bus transactions
- internal multiplexers support up to seven bus masters (ampliable)
- internal arbiter resolve bus contention through round robin policy

SystemC modules (masters and slaves)
- complete SystemC description (lower abstraction level)
- encapsulation of C/C++ code (higher abstraction level)
**System architecture – Processing module**

**Processing module (master)**
- include CPU, cache memory and peripherals (C++ class derived from open source SWARM simulator)
- a wrapper realizes the interface and synchronization layer between the instruction simulator and the SystemC simulation framework
System architecture – AMBA AHB BUS modules

AHB I/F master module

- mast
- address
- hwdata
- hreq
- hready
- hgrant

AHB I/F slave module

- haddr
- hwdataout
- hsel
- hmaster
- hreq
- ctrl_sign
- hready
- readdata
- ready

AHB mux - arbiter – decoder module

Address and control mux

- mast[
- hmaster
- address[

Write data mux

- mast[
- hmaster
- hwdata[
- selector

Read data mux

- hsel[
- hrdata
- hready
- readdata[

arbiter

- hreq[
- hready
- hgrant[
- hmaster
- ctrl_sign

arbiter

- haddr
- hwdataout
- hsel
- hmaster
- hgrant
- hready
- readdata
- ready

decoder

- hsel
- hready
- readdata
System architecture – Test configuration

Shared memory dual processor – dual memory system
//instantiation of the first processing module
armsystem soc0("Arm_System0");
    soc0.clock(clock);
    soc0.pinoutdef(armmaster[0]);
    soc0.readymaster(readymast[0]);
    soc0.requestmaster(requestmast[0]);
    soc0.burst_ready(burstready[0]);
//instantiation of the second processing module
armsystem soc1("Arm_System1");
    soc1.clock(clock);
    soc1.pinoutdef(armmaster[1]);
    soc1.readymaster(readymast[1]);
    soc1.requestmaster(requestmast[1]);
    soc1.burst_ready(burstready[1]);

//instantiation of the first memory module
ram Memory0("RAM_Memory0");
    Memory.clock(clock);
    Memory.rampin(memoryslave[0]);
    Memory.readymem(readyram[0]);
    Memory.requestmem(requestram[0]);
//instantiation of the second memory module
ram Memory1("RAM_Memory1");
    Memory1.clock(clock);
    Memory1.rampin(sott2ram);
    Memory1.readymem(sottfwdready);
    Memory1.requestmem(sottdlyrequest);
Simulator Test – Processing units and bus contentions

Screen capture of our dual processor simulator booting multiple parallel UClinux kernels. The two CPU work independently, contending the bus for memory access.
Simulator Test – Parallel Matrix Multiplication

Application:
Shared memory matrix multiplication executed by two parallel algorithm running on separate processors

Collateral developing activities and tests:
- Manual binding at compilation time of program functions to each processor
- Dual processor startup and initialization code
- Dual processor linker script
Simulation - AMBA burst access timings example

Through SystemC tracing capabilities, cycle accurate signal activity can be visualized.
We have implemented a multiprocessor simulation environment in SystemC 1.0, containing an AMBA bus model, along with masters (CPUs) and slaves (memories) SystemC modules. The AMBA bus model itself is composed by several SystemC modules (arbiter, decoders, multiplexers). AMBA master devices (processors) are represented by instruction-level models of cached ARM cores.

- **Characteristics**
  - Easily scalable through multiple instantiation of modules
  - Easily expandable through new modules implementation
  - Standard and well defined interface for modules interoperability

- **Future works**
  - Multiprocessor operating system, currently under development (RTEMS)
  - Expansion of the simulator with new master and slave modules
System Architecture – Future Expansions

- CPU (master #1)
- CPU (master #2)
- CPU (master #3)
- DMA controller (master #4)
- Memory (slave #1)
- Memory (slave #2)
- I/O (slave #3)
- AHB/APB bridge (slave #4)