ArchC: A SystemC Based Architecture Description Language

S. Rigo, G. Araujo, R. Azevedo and P. Centoducatte
{srgio, guido, rodolfo, ducatte} @ ic.unicamp.br

Speaker: Rodolfo Azevedo

Computer Systems Laboratory
IC-UNICAMP
Overview

• ArchC
• SystemC TLM integration in ArchC simulators
• ArchC Reference MPSoC Framework
• Demo: A dual processor application
• Demo: A videogame console
ArchC Design Flow

Architecture, Instruction set and behaviors

Target CPU description (ADL)

Changes in current CPU or choice of a new one

Tool generator

Requirements met?

Application code

Compiler *

Assembler

Link editor

Simulator

* Under development
AC_ARCH(mips1) {
    ac_mem MEM:32M;
    ac_wordsize 32;
    ac_regbank RB:32;
    ARCH_CTOR(mips1) {
        ac_isa("mips1_isa.ac");
        set_endian("big");
    }
}

AC_ISA(mips1) {
    ac_format Type_R="%op:6 %rs:5 %rt:5 %rd:5 %shamt:5 %func:6";
    ac_instr <Type_R> add;
...
}

mips1_isa.ac

ac_behavior(add) {
    RB[rd] = RB[rs] + RB[rt];
}

mips1_isa.cpp
Main Features in ArchC

- Interpreted simulator
  - Functional or cycle-accurate
  - GDB interface
- Compiled simulator
  - up to 200 Million instructions/s
- Linux OS call emulation
- Runs binary code straight from GCC
- A large pool of real programs successfully tested (MiBench and Mediabench)
SystemC TLM integration in ArchC

• Uses the SystemC TLM library infrastructure
• `ac_tlm_ports`: initiator (master) ports declared on a processor model, used to access
  – External memory
  – Memory Mapped IO or IO instructions
• `ac_tlm_intr_ports`: slave ports declared on a processor model, used for interrupts
• Uses a protocol based on simple request and response packets
SystemC TLM integration in ArchC

User Layer

```
out_port.read(buf, rs, 32);
```

Protocol Layer

```
req.type = READ;
req.address = addr;
req.size = sz;
```

Transfer Layer

```
rsp = (*this)->transport(req);
```

ArchC Instruction behavior

User Layer

```
this->read(&data, req.address, 32);
```

Protocol Layer

```
switch (req.type)
{
  case READ: ...
```

Transfer Layer

```
ac_tlm_rsp
shr_mem::transport(ac_tlm_req& req)
{...}
```
ArchC MPSoC Reference Framework

- **ARP**
  - **Platforms**
    - Framework description and main files
      - Makefiles, software-processor bindings, etc.
  - **Processors**
    - Models and ArchC 2.0 generated processors
      - SPARC, MIPS, ARM, x86, PowerPC, etc.
  - **SW**
    - Cross-compiled software to processors
      - SPLASH test suite
  - **IS**
    - Interconnection Structures Models
      - AMBA, CoreConnect, NoC, PlainBus, etc.
  - **IP**
    - IPs descriptions
      - Video, Keyboard, UART, etc.
  - **Wrappers**
    - Adapters to IPs and ISs
      - TLM-TLM, TLM-RTL, TLM-C++, etc.
Simple Demo

GDB <-> proc1 <-> proc2 <-> shr_mem
Demo – ArchC Declaration

AC_ARCH(mips1) {
    ac_tlm_port out_port;
    ac_regbank RB:32;
    ac_wordsize 32;
    ...
}

//!Instruction lw behavior method.
void ac_behavior( lw )
{
    RB.write(rt, out_port.read(RB.read(rs)+ imm));
};
Demo – MPSoC Instantiation

```c
#include <systemc.h>
#include "mips1.H"
#include "shr_mem.H"
int sc_main(int ac, char *av[])
{
    shr_mem mem("mem", 5242880); /// Shared memory
    mips1 mips1_proc1("proc1"); /// Processor modules
    mips1 mips1_proc2("proc2");

    mips1_proc1.out_port(mem.target_export); /// Bindings to shared memory
    mips1_proc2.out_port(mem.target_export);

    mips1_proc1.load("prog1"); /// Program loading and processor
    mips1_proc2.load("prog2");
    mips1_port1.enable_gdb(5000); /// GDB port configuration

    mips1_proc1.init(); /// Initialization
    mips1_proc2.init();

    sc_start(-1); /// Start simulation

    return EXIT_SUCCESS;
}
```
Demo: videogame console

- RTL
  - NoC
- Functional
  - Processor
  - Video
  - Input
  - Output
Demo: videogame console
Availability

Download NOW!

March 17th

• ArchC 1.6
• Processors:
  – MIPS, SPARCv8, PowerPC, 8051, PIC
• Assembler generator
• GDB interface
• Compiled Simulator
  – Up to 200M instr./s
• Memory hierarchy
• Pre-compiled programs
  – MiBench and MediaBench

• ArchC 2.0 beta 1
• ARP 1.0 beta 1
• ARM processor
• Backward compatible
• TLM integration
• Multicore simulation

Demos in my notebook
Testimonials

“\[I'm \ currently\ doing\ architectural\ studies\ for\ embedded\ processors\ doing\ cryptography.\ ArchC\ turned\ out\ to\ be\ a\ fabulous\ tool,\ ....\]”

Jacques Fournier
Computer Laboratory
Internet

www.archc.org

Documentation, download, bug report, discussion forum

22,817 hits
8:30 am, Wednesday 02/15/06
ArchC MPSoC Reference Framework

• Easy to use framework
  – Autotools compatible (autoconf, automake)
  – Straightforward processor generation (ADL)

• High reuse of IPs and IS
  – User can easily add SystemC IPs and cores
  – Use of wrappers to mix descriptions

• Fast MPSoC creation
  – SystemC ability of refinement
  – Early test phase (both software and hardware)
Main Features in ArchC

• Seamless SystemC integration
• Interpreted simulator
  – Functional or cycle-accurate
  – GDB interface
• Compiled simulator
  – up to 200 Million instructions/s
• Memory Hierarchy modeling
• Linux OS call emulation
• Runs binary code straight from GCC
• A large pool of real programs successfully tested
• 15+ researchers involved for the last 3 years
• Very happy users !!!
MPSoC Design in SystemC

• ESL design space exploration
  – Large amount of different physical processors available
  – Time-to-market
  – Need tools and high level modeling to speed up design process!!!

• Why an ADL?
  – Processor designer x Processor user