Fast Simulation of SystemC Designs with Scoot

Nicolas Blanc, Daniel Kroening

www.cprover.org/scoot

Supported by Intel and SRC
Theme of the Presentation

Utilization of the semantics of SystemC for Simulation speedup.

- System Description Language
- Based C++
- Compilation using g++
- IEEE Std.

SYSTEM C

- Compiler for SystemC
  - www.cprover.org/scoot
Execution Time (Oct 09, Linux 3Ghz, gcc 4.2.4, Linux)
Simulation Speedup

![Bar chart showing speedup for different benchmarks](image-url)
Outline

- Overview of SystemC
- Overview of Scoot
- Demo AES128
System Description Languages

**C++**
(High-level software view)

- Software Model
- Transaction Level
- Semantics Gap

**Verilog**
(Low-Level hardware impl.)

- Behavioral Level
- Register -Transfer Level
- Gate Level

New System Description Languages:
- SystemC
- SystemVerilog

2000
SystemC versus SystemVerilog

- **SystemC**
  - Hardware
  - SystemC

- **SystemVerilog**
  - Verilog Extension
  - Verilog
  - C++ Library
  - C++
Example: Memory Module

```cpp
#include <systemc.h>

SC_MODULE (Memory) {
    sc_in<bool>           clk, cs, we, re;
    sc_in<sc_uint<32>  > data_in;
    sc_in<sc_uint<32>  > addr1, addr2;
    sc_out<sc_uint<32> > data_out;

    SC_CTOR(Memory){
        SC_METHOD(read()); senstive << clk.pos();
        SC_METHOD(write()); senstive << clk.pos();
    }

    void read();
    void write();
};
```

Inherits from `sc_module`
Module Interface
Module Constructor
Processes
```c
int sc_main(int argc, char* argv[]) {
    sc_clock clk;
    sc_signal<bool> cs, re, we;
    ...

    Memory mem("MEMORY");
    Testbench tb("TEST BENCH");

    mem.clk(clk);
    mem.cs(cs); mem.re(re); mem.we(we);
    ...

    sc_start(10, SC_US);
    return 0;
}

SystemC Simulation:
g++ main.cpp memory.cpp tb.cpp -lsystemc -o simulator
```
The Concurrency Model

- Execution driven by events
- Cooperative Multitasking Model:
  - Only one process running at a time
  - **No preemption!**
The SystemC Scheduler

Begin

Evaluation Phase

Update Phase

Delta Notification

Time++

Time Notification

End
Observations

- Elaboration of the Module Hierarchy at Runtime:
  - Modules, processes, port binding,... The approach is flexible!

- C++ is fast: Fast execution of the processes!

- Yes ... but, GCC is not taking advantage of SystemC information!
  - module hierarchy, processes, and port binding.
Overview of Scoot

- Scoot statically discovers:
  - Module hierarchy, port binding, processes, and sensitivity lists.

- Simulation benefits from:
  - Resolution of dynamic calls (static-scheduling)
  - Suppression of dynamic data structures in the scheduler (lists, sets).
  - Propagation of port binding information (pointers).
Scoot

Typechecker (C++ frontend)

Control-Flow Graph

Pointer Analysis

SystemC Analysis

Static Scheduling

Code Re-Synthesis

Flat C++ Model

SIMULATOR

In-house development

Path and field sensitive

Generates C++ code

Rely on Pointer Analysis

Simplified SYSTEM C Library

C++ Files

Monday, 05 October 2009
Department of Computer Science
Demo

Benchmark: Encrypt, decrypt, and then display 128-bit vectors. Simulation Time: 800 Microseconds.
Conclusion

- Elaboration of the Module Hierarchy at Compile Time:
  - We sacrifice some flexibility in exchange for
  - significant simulation speedup, and
  - we can now reason about SystemC models statically!
    - Formal Verification, e.g., previous talk about static race analysis.

Thank You!