A Framework for the Analysis of SystemC Descriptions

Franco Fummi
3rd European SystemC Users Group Meeting

Dipartimento di Informatica
Università di Verona

Sitek S.p.A.
Verona
Agenda

- Motivations
- Framework Structure
- Developed Modules:
  - VHDL to SystemC Converter
  - Functional Test Generator
- Scheduled Extensions
Motivations

➢ Tools manipulating SystemC code do not require:
  ➢ syntactic analysis (C++ compiler)
  ➢ code simulation (SystemC simulation kernel)

➢ they require:
  ➢ identification of models
  ➢ interconnection
  ➢ identification of SystemC keywords (before C++ preprocessor macro expansion)
  ➢ creation of CDFG for each model

Code access after removal of syntactic sugar
Framework Starting Point

- A commercial proprietary tool:
  - no one available for research purposes

- A public domain C++ compiler (e.g., g++):
  - complete syntax of C++
  - optimized Abstract Syntax Tree (AST)
  - macros are not preserved
  - difficult access to internal structure
  - basic syntactic elements not grouped
Framework Starting Point

- A public domain HDL analyzer (e.g., VHDL):
  - different syntax ⇒ modification of the syntactic analyzer
  - available methods for internal data structures creation
  - efficient debugging capabilities

Is the internal data structure adapt for SystemC?

Yes for hw descriptions
No for general descriptions
Selected Framework

- **Savant** (University of Cincinnati)
  Standard Analyzer of VHDL Applications for Next-generation Technology
  - VHDL analyzer
  - Internal Intermediate Representation (IIR)
  - C++ translator (in-code simulation kernel)

- **Choice reasons:**
  - VHDL parser built by using PCCTS
  - simple extension of the IIR data structure
Framework Structure

- VHDL code
- SystemC code
- C++ code
- Code Transformation
- Savant
- Parser
- IIR
- VHDL code
- SystemC code
VHDL to SystemC

- Entity/Architecture
- Port declaration
- Signal declaration
- Variable declaration
- Logic/arithmetic operation
- Assignment
- Procedure, function
- Process
- Control operation
- Package
- Type/subtype/array
- Event attribute

- Generic
- Component
- Record
- Resolved
- Attribute
Functional TPG

Key idea:
- both system description and TPG are written in C++ and jointly compiled $\Rightarrow$ high efficiency

Characteristics:
- Independent on the description language
- Test generation based on a error model
- Independent on test generation technique
- Some parts are composed of SystemC code:
  - injected errors = code transformation
  - module selection = code transformation
  - erroneous/error-free comparison = ad-hoc code
TPG Architecture

- VHDL/ SystemC code
- SystemC code with errors

Error Injector
- VHDL code with errors
- Error List

TPG Generator
- SystemC code for Comparator
- C++ TPG Controller

Control Point Injector
- Controllable/ Observable SystemC code with errors
On-going Tasks

- CLP (constraint logic programming):
  - SystemC to CLP for code analysis
- AI (abstract interpretation):
  - SystemC to AI for code analysis
- Parametric Code Generation
  - SystemC configurable cores

Tasks Coordination through

*names space*

programming approach
Scheduled Extensions

- VHDL to SystemC translation:
  ✓ 2\textsuperscript{nd} quarter 2001

- Functional TPG and error simulator:
  ✓ 2\textsuperscript{nd} quarter 2001

- Prototypal version of SystemC to IIR:
  ✓ 3\textsuperscript{rd} quarter 2001

- SystemC to IIR compilation:
  ✓ 4\textsuperscript{th} quarter 2001
To Cooperate

ED@ Lab.

Electronic Design Automation meets WEB-based Technologies

Latest modifications:
- The site map is visible with the button on the top of this site.
- The contact us link is under construction; for contact us you find our e-mail address under this page.
- Search is available.

http://eda.sci.univr.it

Franco Fummi