RTL2ASC
Description of the Functionality

Richard Buchmann and Alain Greiner
E-mail: richard.buchmann@lip6.fr, alain.greiner@lip6.fr

1 RTL2ASC Overview

RTL2ASC is a **SystemC** model generator. That tool converts any VHDL source code at RTL level to a SystemC code. Generated models are described as a set of synchronous finite state machine and are cycle accurate. Its goal is to accelerate the hardware simulation by allowing cycle based simulation. Efficiency is obtained through the use of the Finite State Machine description (Figure 1).

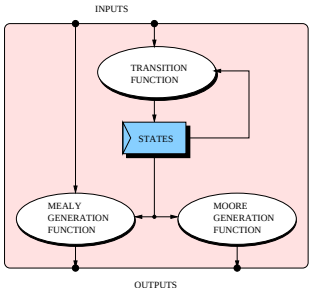

Figure 1. Finite State Machine Modeling

RTL2ASC inputs are in a RTL language subset. This subset is fully compatible with the IEEE VHDL'87.

RTL2ASC output is a SystemC subset including :

- core class :
  sc_module, sc_signal, sc_in, sc_out, sc_inout...
- basic data type :
  sc_uint only

Generated SystemC source code uses only SC_METHOD process type. That source code runs up to 10x times faster on using SystemC-2.1 and SystemCASS[1] simulators.

2 Installation

Copy RTL2ASC files, then run the Makefile.

The distribution is as follow :

- *src* : Source files
- *bin* : Binaries for linux distribution
- *examples* :
  - *miscellaneous test benchs* : Some simple hardwares to test some very basic VHDL constructions.
  - *HADAMARD sequencer* : Finite state machine to drive Hadamard data path.

3 Generator Execution

To use RTL2ASC generator, you just need to execute the binary as follow :
../bin/rtl2asc [-h] [-s] [-d] [-f] [-x] <source without file extension>

- h : display help
- v : verbose mode
- d : save data dependencies into dot file
- f : drive rtlfig (ALLIANCE structure)
- x : save structures into XML file
- s : drive to SystemC model

4 A Semantic Approach

RTL2ASC uses a semantic analysis to convert from RTL to cycle accurate abstraction level.

Front-End part reads source input and builds up the internal representation. The current front-end reads only RTL VHDL code.

The internal representation allow us to describe the parallel execution of the set of processes, and signal assignments. So, processes are compiled into control graphs[2]. Control graph relies on Petri nets.

We apply some transformations to get a reduced control graph. This graph allows us to extract a finite state machine with a set of unique assignments.

Back-End part writes the finite state machine into files. The current back-end writes only SystemC code.

References

[1] R. Buchmann, F. Petrot, and A. Greiner. Fast cycle accurate simulator to simulate event-driven behavior. In *Proceeding of The 2004 International Conference on Electrical, Electronic and Computer Engineering (ICEEC'04)*, pages 35–39, Cairo, Egypt, 2004. ASIM/LIP6, IEEE.

[2] Bawa Rajesh K. Jacomme Ludovic, Pétrot Frédéric. Formal analysis of single wait vhdl processes for semantic based synthesis. In *12th IEEE International Conference on VLSI Design*, pages 151–156, 1999.