A2T++: the Automatic Abstraction Tool

Nicola Bombieri  Franco Fummi  Graziano Pravadelli
ESD Group
{firstname.lastname}@univr.it
University of Verona, Italy
http://esd.sci.univr.it/en.html

Abstract

IP-reuse allows designers to exploit already implemented and verified RTL IP cores while concentrating the main effort on their integration into the system, on other specific components implementation and on the SW development. In this context, this work presents A2T++, a tool for abstracting existing RTL IP cores aiming at two different targets: (i) automatic generation of SystemC TLM models, and (ii) automatic generation of embedded software for multiprocessor architectures. A2T++ is built on top of the HIFSuite, which allows designers to manipulate and integrate heterogeneous components implemented by using different HDLs, that are SystemC, VHDL, and Verilog.

1. Introduction

Reuse of previously-developed IP cores is the key strategy for facing the increasing pressure in design productivity. Prevailing IP-reuse methodologies are based on sharing designs typically implemented at RTL level (i.e., soft IP-reuse) in a hardware description language (HDL) such as VHDL and Verilog [1].

In the context of transaction-level modeling (TLM), designers rely on two different solutions for reusing RTL IPs: (i) reusing RTL IPs via transactor, and (ii) abstracting RTL IPs into TLM implementations. In the first solution, the transactor is exploited for allowing simulation and verification of TLM-RTL mixed designs [2]. Nevertheless, reusing RTL IPs via transactor has two main problems: firstly, transactor implementation is actually a complex and error-prone task and verifying its correctness often needs more effort than verifying the RTL IP integration. The second problem concerns the simulation performance, since the RTL component acts as bottleneck into the mixed level design, thus slowing down the simulation of the whole system.

As a first outcome, this work presents A2T++ to automatically abstract RTL IPs into equivalent TLM descriptions (left side of Figure 1), thus allowing simulation to maintain the high performance proper of TLM. A2T++ relies on the standard SystemC TLM-2.0 library proposed by OSCI [3] for guaranteeing interoperability of the generated TLM models.

On the other hand, performance improvement of multiprocessor architectures often induces designers to choose a processor to execute an application program rather than using dedicated hardware. In this context, it should be often more convenient to translate existent RTL IP descriptions, originally dedicated to implement HW components, into pure SW code (i.e., C/C++) in order to exploit the advantages of the multiprocessor architectures [4].

The second outcome of this work is to present A2T++ to automatically generates SW code (i.e., C++) by abstracting existent RTL IPs implemented in HDL (right side of Figure 1). A2T++ exploits an abstraction algorithm for eliminating many implementation details typical of the HW descriptions, in order to improve the performance of the generated code.

2. Automatic generation of SystemC TLM models

The methodology implemented in A2T++ to abstract RTL IPs into equivalent TLM models is composed of the following main steps [5]:

1. Generation of TLM functionality. Firstly, different computational phases composing the behavior of the RTL IP are identified. They consist of an input sub-phase, where the IP gets data from the primary Inputs, an elaboration sub-phase, where the IP computation is carried out, and an output sub-phase, where the computation result is put to the primary outputs. Every sub-phase is composed of a sequence
of adjacent states, when the IP is modeled like an FSM. Then, all the states composing each different input/elaboration/output sub-phases are merged into macro states. Finally, the clock is removed and the IP becomes an entity accurate on transaction events (see Figure 2).

2. Generation of TLM communication protocol. The communication protocol side of the TLM IP is generated by implementing an interface compliant with a TLM standard library (i.e., standard OSCI TLM 2.0 [3]), in order to guarantee the generated model interoperability and reuse. Moreover, an opportune TLM IP driver is implemented to allow any master to correctly interface to the TLM IP. The driver is automatically implemented by analyzing the input/elaboration/output sub-phases of the RTL IP description.

3. Automatic generation of embedded software for multiprocessor architectures

Figure 3 depicts the main phases of the methodology implemented in A2T to automatically generate C++ code from RTL IPs implemented in HDL code. The abstraction algorithm consists of four main steps:

1. The starting point. The RTL IP code is parsed and the extended FSMs (EFSMs) composing the processes of the model are extracted (Figure 3.a). The EFSM model can represent any RTL description.

2. Merge of processes. The processes are merged in order to compose a single main process (Figure 3.b). This phase does not suffer of any state-explosion problem.

3. Merge of EFSM states and interface definition. The main process is abstracted into a pure C++ function, by collapsing the states into macro states. The C++ function interface is defined by generating an I/O structure passed as argument of the main function (Figure 3.c).

4. Definition of communication protocol. The protocol definition consists of defining the right sequence of values that an external actor (i.e., the master) has to settle in the I/O structure, in order to correctly access to the IP functionality.

A2T++ is built on top of the HIFSuite [6], which allows designers to manipulate and integrate heterogeneous components implemented by using different HDLs, that are SystemC, VHDL, and Verilog.

References


