The Use of SystemC for Design Verification and Integration Test of IP-Cores

Alessandro Fin Franco Fummi Denis Signoretto
DST Informatica, Università di Verona, ITALY

ABSTRACT

The current trend of systems on silicon is leading to System on Chips with embedded software and hardware components. Design simplification is becoming necessary to respect the target time-to-market of SoCs, and this goal can be obtained by using predesigned IP-cores. However, their correct integration in a design implies more complex verification problems. The SystemC language allows to create and integrate accurate models of software algorithms, hardware architectures and interfaces for SoCs. In this paper, characteristics of the language are exploited to define a design verification framework for integration test of IP-cores. Intellectual property of cores is guaranteed by adopting a client/server simulation architecture and by allowing functional test generation on faulty IP-core models without disclosing their internal structure. Moreover, the methodology can be applied to mixed descriptions based on VHDL and SystemC, since an abstraction layer has been defined allowing clients and/or servers to be indifferently described in VHDL or SystemC.

1. INTRODUCTION

The current trend of systems on silicon is leading to a complexity that can be reduced only by integrating already produced and optimized parts (IP-cores). The identification of the more suited core for a design is one of the more time consuming aspect of design management [1]. In fact, the effective evaluation of the correct integration (*integration test*) of the core can be performed only by running some simulation sessions. Moreover, integration test is a difficult task since functional test patterns must be generated accordingly to the design where the core is embedded. Two main problems can thus be identified:

- **simulation** of IP-cores at different abstraction levels, without disclosing intellectual property, in order to evaluate the most suited core for the selected application;
- **functional test patterns** generation for the integration test of the selected core into the design,

For both reasons, some simulatable views of a core must be available to the core user and they must model the core at different abstraction levels (behavioral, RT, gate, switch). The new hardware-software description language (SystemC) is emerging and it is becoming a new standard in the EDA field. The distribution of a SystemC description is safe from the point of view of the protection of the intellectual property if the used abstraction level hides implementation details. However, this condition cannot be always guaranteed if the evaluation of the core requires detailed simulation sessions and since functional test generation requires faulty models of the core.

Cryptographic techniques for delivering simulation models have been proposed [2, 3], but they do not seem to completely solve the problem, since they are extremely simulator dependent (in the case of VHDL) or computer architecture dependent (in the case of compiled SystemC models). This paper proposes to concentrate the source of simulations directly in the Web server of the core vendor, by proposing a typical client-server architecture, where the core users perform distributed simulations by connecting their simulation environment to the simulation environment of the core vendor. This idea has been already analyzed in the literature [4, 5], by proposing the use of ad-hoc languages, such as Java, to model the core functionality and allowing cooperative working. The main disadvantage of such approaches is the need of a remodeling phase, by the core vendor, which does not usually use Java-based tools to design cores. Even disregarding the necessary extra work, there is a high possibility of introducing discrepancies and differences between the Java models and the design models. An alternative approach [6], centered to VHDL descriptions only, was affected by the intrinsic simulation degradation produced by interfacing a event-driven VHDL simulator with a message-passing transmission protocol. Moreover, the integration test problem cannot be solved by using such an approach.

The first aim of this paper is to analyze the impact of the new design language SystemC on a Web-based simulation methodology to perform high-level integration and validation test of IP cores. By adopting this language, the core user has only to view the core interface (the class declaration in the `header` file) and the responses of the simulator server in relation to each vector submitted. No internal information of the model is discovered, thus preserving the intellectual property. The simulation methodology is based on standard techniques and tools, that is, the SystemC library and simulator, the Internet protocol and a socket-based [7] interface. In this way, mixed VHDL-SystemC simulation sessions can be organized by simply embedding in the design the correct socket interface, moreover, the core user can select the abstract level of the analyzed core by ranging from very abstract system-level (SystemC) descriptions to very accurate gate-level representations (VHDL or Verilog) by using the same simulation environment.

0-7803-6741-3/901/$10.00

The second aim of this paper derives from this simulation framework and it concerns a distributed functional test generator able to solve the integration test problem. It allows functional test generation for SystemC descriptions including IP-cores. Both fault-free and faulty core descriptions are remotely simulated in order to generate functional test patterns useful to perform integration test of the core. The faulty core produces faulty responses accordingly to a generic functional error model selected by the core vendor. No information related to the errors are disclosed, since a simple error number is used to identify an error. In this way, the proposed testing methodology allows to check the correct integration of a core into a larger design, by producing functional test patterns covering errors of both core and the surrounding logic. In a related paper [8] only a proof of concept has been given for the fault simulation of IP-based designs. However, such an approach can be reasonably applicable only to simple combinational circuits, where the number of signals, and hence the traffic on the network, is low.

The rest of the paper is organized as follows. The methodology for distributing IP-cores is presented in Section 2. Section 3 discusses all features and problems related to the simulation of SystemC descriptions across the Internet. Section 4 shows the implementation of a remote test generator for the integration test of a core. An application example is described in Section 5, while Section 6 is devoted to future works and concluding remarks.

2. IP-CORE DISTRIBUTION

The proposed methodologies for IP-Core distribution can be described by considering the point of view of the two *cooperating parts*: the core vendor and the core user.

- The goal of the **core vendor** is to allow the core user to simulate the cores without discovering their internal descriptions. A method to allow this simulation is the use of a client-server architecture [6].
- The **core user** usually analyzes at first the general characteristics of the core reported in the Web site of the core vendor. Results of this rough analysis must then be improved by performing some simulation sessions, eventually remotely, to explore the effective integration of the core in the design. This operation is based on some information downloaded from the Web site of the core vendor.

The main problem of the core vendor is to release core descriptions without disclosing IP information. This paper proposes the following three different methods to solve this problem: two different solutions for performing remote simulation and one solution allowing local simulation.

- **VHDL-C suite**
  This suite is oriented to VHDL designers. It is composed of the core interface (a VHDL entity) and the socket interface (a C-language architecture). This interface does not include information concerning the core functionality, but it implements the communication protocol only. The VHDL simulator adopted by the designer must be able to link C objects to allow the transmission of data packets with simulation values. Both SystemC and VHDL designers are the users of this suite.
- **SystemC suite**
  It is composed of a core interface (a SystemC component declaration) and the socket interface (a SystemC component definition). The SystemC component definition is released as a C++ source file and the core interface as a C++ header file. SystemC designers can simply compile and link this suite to their design in order to remotely simulate the core. Moreover, VHDL designers can also use this suite to verify the integration of the core in their preliminary SystemC-based prototype. When the design will be represented in VHDL, the same suite or the VHDL-C suite will be used to refine the simulation.
- **SystemC suite for local simulation**
  It is oriented to the same users and can be used for the same targets. It is composed of a core interface (a SystemC component declaration) and its implementation (a SystemC component definition). The SystemC component definition is released as a C++ object file, compiled for the different supported computer architectures, and the core interface as a C++ header file. This suite can be directly linked to a SystemC design or interconnected by suing socket to a VHDL design.

All the suites have to be published in the Web pages of the core vendor to allow the interested designers to download the usable one. The last suite has been designed for a local simulation session, while the other suites allow remote simulation. Remote simulation allows the core vendor to maintain and update only one version of the cores, while their distribution requires a more complex upgrading methodology. However, remote simulation decreases the simulation performance required by the core user. Experimental results reported in Session 5 show the relative performance of all approaches.

3. IP-CORE REMOTE SIMULATION

The remote simulation is guaranteed by the client/server architecture, which connects the local simulator to the remote simulation server. All involved parts, from the client and the server side, are reported in Figure 1.

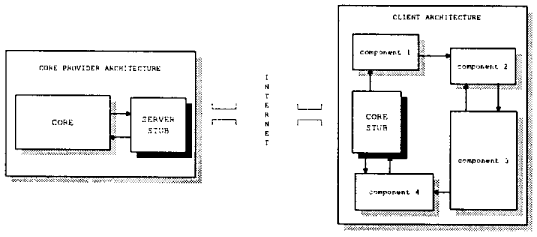

Figure 1: Client/server architecture for remote simulation.

This client/server architecture is based on two stubs: *server stub*, the remote core socket interface, and *core stub*, the local core socket interface. Three different combinations of these stubs have been developed and analyzed.

- **Pure SistemC**
  SystemC descriptions can be adopted for both remote and local simulations. In the case of a remote simulation, there is a simulation server running on a core-vendor computer connected to Internet. The simulation server is based on a SystemC description of the core. The core-user simulator establishes a connection to the simulation server, sending and receiving messages containing port values. This solution is perfectly adapted for core users that design by using SystemC and would like to analyze the integration of a system-level described core. By choosing a local simulation method, the integration of the design and the core can be performed by linking both SystemC descriptions or by interconnecting them via socket. The main drawback of this solution concerns the requirement for the core vendor to have SystemC models of the cores. However, automatic translation of VHDL into SystemC can be performed as mentioned in Section 5.

- **Pure VHDL**
  By adopting this solution, both server and client must have a VHDL simulator able to link C objects in order to implement the socket interface. By using remote simulation, the upgrade of the simulated core description is obtained by simply upgrading the description running on the VHDL simulation server. VHDL allows the simulation of the core at different abstraction levels (behavioral, RT, gate), thus making possible to study the core integration during different developing phases. The main drawback of this approach concerns the degradation of the simulation performance due to the use of the C interface. Moreover, in the case a local simulation is required, VHDL executable versions of the core must be downloaded from the core-vendor Web site. However, such versions are usually very simulation tool dependent, thus making this approach very difficult to be developed and maintained.

- **Mixed architecture**
  This solution is based on a combination of a VHDL *core stub* and a SystemC *server stub*. To develop this solution both stubs have to perform data conversion between the VHDL and SystemC types (e.g., `std_logic` versus `sc_logic`). This solution can be interesting for VHDL designers that would like to analyze the integration of a core described at the system level into their RT-level designs. The SystemC executable version of the core can also be distributed and locally connected via socket to the VHDL user simulator.

The synchronization of the client and server seems to be a non relevant problem, since it is solved by the socket. On the contrary, it is necessary to solve some hidden problems originating from the differences between the semantic of the VHDL simulator (event driven) or SystemC (cycle based) and the semantic of the socket interface (message driven). This analysis is omitted for lack of space.

4. REMOTE TEST GENERATION

The proposed simulation framework can be used to implement a distributed test generator as described in Figure 2. The architecture is composed of the test generation program and some fault-free and faulty modules, which are compared for generating functional test patterns.

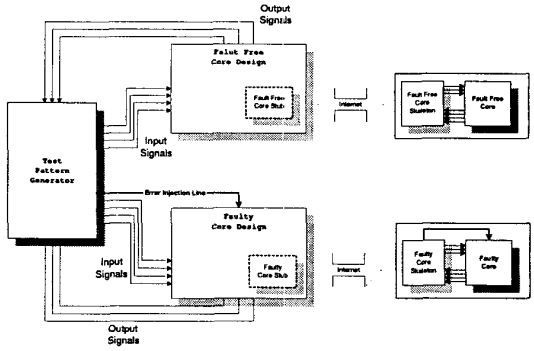

Figure 2: Remote test generator architecture.

The whole test generation procedure is locally performed with the exception for the simulation of the fault-free and faulty cores. In fact, the core is replaced by a core stub. The main task of the stub is to manage the socket interface to communicate with the core vendor server. Test patterns are sent, by the core stub, to the core server. This module applies the stimuli to the real core and send back the output to the TPG. This working mode is the same used for normal remote core simulation.

The faulty core stub has an extra port to select the injected fault. The total number of faulty configurations available is a parameter chosen by the core vendor. Moreover, to obtain a better IP protection, error codes are not correlated, i.e. two errors on bit 6 and 7 of the same core signal are coded by x and y, such that $y \neq x + 1$.

The absence of information about the internal structure of the core does not allow the core user to apply deterministic algorithms for error detection. Thus the current release of the TPG is based on a random test pattern generator. This simple approach is currently extending by using probabilistic algorithms, such as genetic algorithms.

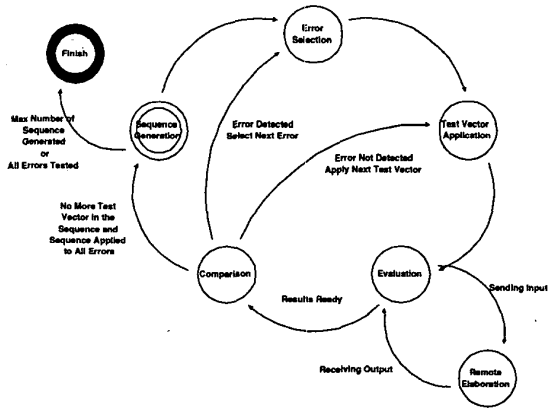

Figure 3: TPG diagram.

The test pattern generation process is designed as a finite state machine (Figure 3) composed of the following states: *test sequence generation, error selection, test vector application, evaluation, remote elaboration,* and *comparison.* When the evaluation state is reached for the core test, the local computation stops, and the input and the code error are sent, by the socket interface, to the remote server. The execution restarts as soon the outputs are received from the core vendor server and it continues with the comparing phase.

5. APPLICATION EXAMPLE

The proposed methodology for the verification and integration test of IP cores is applied in this section to an example. The Modeltechnology VHDL simulation environment has been adopted for the VHDL descriptions, since it allows the mixed simulation of VHDL and C modules, while SystemC 1.1 has been used for modeling and simulating the SystemC descriptions. All experiments have been performed on a SUN Ultra5 333 MHz with 256MByte Ram.

5.1 Core Selection

We consider the problem to implement a 3-tap digital filter characterized by the following equation:

$$y = c_0 z + c_1 z^{-1} + c_2 z^{-2}$$

The hardware realization of the filter uses, as embedded core, a public domain load/store CPU available at the RT and logic levels. The SystemC (VHDL) definition of the core is provided on the Web site of the core vendor with the *core stub* and the corresponding implementation . They allow the remote simulation of the core.

5.2 Core Simulation

The SystemC (VHDL) simulation of the core is performed to verify the effectiveness of the selected core. Its correct integration in the design is checked by generating functional test patterns as reported in the next paragraph.

We performed four types of simulation to measure the applicability of the proposed simulation methodology:

- **Local** simulation of the core embedded into the global architecture. It corresponds to the simulation of a core, which is directly provided in SystemC (VHDL) source code.
- **Local with socket** simulation. Both client and server are running on the same machine and they are interfaced via socket. This simulation measure the overhead of the socket interface disregarding network problems.
- **Intranet** remote simulation. Both client and server belong to the same Internet domain and they are connected trough a 100Mbit Ethernet connection. This simulation is related to the use and distribution of a core in the same company, where a design group can only use the results of another design group without disclosing the intellectual property.
- **Internet** remote simulation. This is the more general case, where a local client is connected to a remote server located in any part of the Internet. We simulate a traffic condition close to have a client in Europe and a server in north America, or vice versa.

The previously described four types of simulation have been applied to the three different combinations of server and client listed in Section 3.

Table 1 shows the time (in seconds) necessary to run the simulation of the core and the normalized simulation times with respect to the times of the local simulation. Table 2 reports the simulation times normalized with respect to the SystemC local simulation time.

By analyzing Table 1 it is possible to identify the impact of the socket interface and the communication channel on SystemC and VHDL simulators. The simulation performance of the SystemC client/server architecture is affected in the same manner by the two factors. In fact, there is a degradation in simulation time of one order of magnitude due to the introduction of the socket interface and another order of magnitude due to the Internet communication channel. In the case of the VHDL client/server architecture, performance is decreased of two orders of magnitude by simply adopting the socket interface. The use of communication channels does not relevantly modify times. The mixed configuration, VHDL client and SystemC server, shows simulation times comparable with those of the VHDL simulation. This result is expected, since a cooperative client/server architecture works at the speed of the slower peer. Table 2 shows the performance improvement reachable by adopting SystemC (for client and server) instead of VHDL. The SystemC local simulation is 44% faster than the equivalent VHDL simulation. This is promising for the extensive use of SystemC for simulating complex embedded systems. The Intranet SystemC simulation is 8 times faster than the corresponding VHDL simulation, thus emphasizing the more efficient link of SystemC with other libraries with respect to VHDL.

5.3 Core Integration Test

As shown in Table 2, the pure SystemC architecture is the faster solution and hence the best candidate for testing and verifying the correct integration of a remote IP-core within a larger design. To perform integration test, we applied the functional test generator described in Section 4.

We assume that the core vendor distributes both fault-free and faulty CPU core through the previously described remote interface. This guarantees the distribution of the IP--core without disclosing its intellectual property. As shown in Figure 2, we connected the system with a test pattern generator that allows to run a simulation to verify the functional correctness of the global design, thus producing test patterns to check the correct integration of the core. As reported in Table 3, the main factor that decreases the test generation performance is not the test generation phase itself and the socket interface, but the remote simulation. The slowdown obtained for the Internet solution is of two orders of magnitude higher than the local solution. If remote simulation is considered feasible, remote test generation becomes feasible.

6. CONCLUDING REMARKS

A methodology for IP-Core analysis and simulation has been presented . It allows the core vendor to make available very detailed core models without disclosing IP information.

Sim. Arch. Type	Sim. type	Real Time	User Time	System Time	Norm. Real	Norm. User	Norm. System
	Local	6.52	5.82	0.55	1	1	1
Server: SystemC	Local+socket	102.41	3.60	3.41	15.70	0.62	6.20
Client: SystemC	Intranet	104.24	3.91	3.62	15.98	0.67	6.58
	Internet	759.96	4.13	4.10	116.55	0.71	7.45
	Local	9.37	6.78	1.14	1	1	1
Server: VHDL	Local+socket	852.08	5.32	3.80	90.94	0.78	3.33
Client: VHDL	Intranet	857.35	3.41	2.24	91.50	0.50	1.96
	Internet	901.80	3.65	2.67	96.24	0.54	2.34
	Local	-	-	-	-	-	-
Server: SystemC	Local+socket	857.05	3.27	1.61	1	1	1
Client: VHDL	Intranet	857.15	4.26	1.67	1.00	1.30	1.04
	Internet	903.31	4.53	2.12	1.05	1.39	1.32

Table 1: Simulation times (*Real, User and System*) and normalized times with respect to the local simulation time.

Simulation Server Type	Simulation type	Real Time	User Time	System Time
	Local	1	1	1
Server: SystemC	Local+socket	15.70	0.62	6.20
Client: SystemC	Intranet	15.98	0.67	6.58
	Internet	116.55	0.71	7.45
	Local	1.44	1.16	2.07
Server: VHDL	Local+socket	130.67	0.91	6.91
Client: VHDL	Intranet	131.49	0.59	4.07
	Internet	138.31	0.63	4.85

Table 2: Normalized simulation times for SystemC versus VHDL comparison.

Sim. Arch. Type	Sim. type	Real Time	User Time	System Time	Norm. Real	Norm. User	Norm. System
	Local	4.422s	3.340s	0.190s	1	1	1
Server: SystemC	Local+socket	20.452s	5.040s	4.370s	4.625	1.510	23
Client: SystemC	Intranet	21.513s	4.770s	2.810s	4.865	1.428	14.79
	Internet	675.474s	4.970s	3.360s	152.753	1.488	17.68

Table 3: Test generation times (*Real, User and System*) for SystemC suite.

Moreover, it allows the core user to perform a validation test to verify the correct integration of the selected IP core into the core-based design under development. This is performed by generating functional test patterns for both the core and the surrounding logic. This main idea has been exploited by using two different modeling languages: VHDL and SystemC. Experimental results showed that, the use of SystemC produces better results in terms of simulation time and SystemC models can be more efficiently linked to interface libraries. Performance degradation of the remote test generator is dominated by the remote simulation process, thus it seems to be feasible whenever remote simulation is acceptable. Future work will be devoted in the definition of a more efficient data transmission protocol in order to decrease the impact of Internet on the simulation and test generation time. However, waiting for an improvement of Internet transmission capabilities, the proposed technique can be directly used in the contest of the SystemC suite for local simulation, by providing detailed descriptions of the cores, even with faults, without disclosing IP information.

7. REFERENCES

[1] J. Notbauer, T. Albrecht, G. Niedrist and S. Rohringer. Verification and management of a multimillion-gate embedded core design. *Proc. ACM/IEEE DAC*, pages 425–428, 1999.

[2] M.J. Silva and R.H. Katz. The case for design using the World Wide Web. *Proc. ACM/IEEE DAC*, pages 579–585, 1995.

[3] S. Hauck and S. Knoll. Data security for Web-based CAD. *Proc. ACM/IEEE DAC*, pages 788–793, 1998.

[4] R. HelaiHl and K. Olukotun. Java as a Specification Language for Hardware-Software Systems. *Proc. IEEE ICCAD*, pages 690–697, 1997.

[5] M. Dalpasso, A. Bogliolo and L. Benini. Specification and Validation of distributed IP-based designs with JavaCAD. *Proc. IEEE DATE*, pages 684–688, 1999.

[6] A. Fin and F. Fummi. A Web-CAD Methodology for IP-Core Analysis and Simulation. *Proc. ACM/IEEE DAC*, pages 128–133, 2000.

[7] L. Peterson and B. Davie. Computer Networks: A System Approach. *Morgan Kaufmann*, 1996.

[8] M. Dalpasso, A. Bogliolo, L. Benini and M. Favalli. Virtual Fault Simulation of Distributed IP-based Designs, *Proc. DATE*, pages 99–103, 2000.