41 SOCP2P: A PEER-TO-PEER IPS BASED SOC DESIGN AND SIMULATION TOOL

Samy Meftali and Jean-Luc Dekeyser
Laboratoire d'Informatique Fondamentale de Lille. FRANCE
Meftali@lifl.fr, Dekeyser@lifl.fr

In this paper, we present a novel approach and tool allowing the use of the Peer-to-Peer concept in the IP based SoC design and simulation. This tool permits a real collaboration in SoCs design between different teams, and consequently may reduces dramatically the time-to-market. The IPs may be described using different languages. They are automatically encapsulated in a SystemC wrapper communicating with a CORBA transport object. The use of CORBA bus for the communication provides the tool a good flexibility and allows languages heterogeneity.

1. INTRODUCTION

Now a days SoCs are more and more complex and integrate software parts as well as specific hardware parts (IPs "Intellectual Properties" generally). Due to this complexity, the time spent to verify and validate these systems is continuously increasing and may be usually greater than half of the time-to-market of a system.

Modern SoCs are usually designed by assembling some existing hardware or software components called IPs. These later may be available as descriptions in a specific language at one or several abstraction levels.

Because of the slowness of IPs exchange between providers and consumers (designers), and the strong time-to-market constraints related to any modern design projects, all the major design companies and some academic research laboratories have their own IP libraries. However, for cost and competences reasons, no one of these libraries can cover all the IPs which may be used in all the design projects of a given company.

To face all these problems related to IPs exchange (slowness of the exchange procedure, cost, time to market,…), usually providers accept to deliver the IPs interfaces (without any behavior source code) to possible consumers. To take advantage of this possibility, designers must have a tool allowing them to simulate their system just by using the furnished IPs interfaces and without waiting until the end of the administrative IP transaction procedure.

This means that this tool has to be able to simulate a SoC composed by IPs available in the local host of the designer and some distant IPs available on the provider's hosts. For confidentiality reasons, these distant IPs have to be simulated on the provider's host, without downloading them.

This problem is closely similar to the free files and tools exchange on Internet. In fact, we can imagine that the exchange of the IPs interfaces and the simulation authorizations can be done using a tool similar to Napster (Napster) or eDonkey (eDonkey2000, 2000).

In addition to the time-to-market reduction with the anticipation of final IP transaction, this tool can provide many other advantages.

In fact, communication means become increasingly powerful. So it becomes more and more interesting for designers to have tools allowing them to simulate systems on distributed platforms (some IPs on local hosts, and other IPs on the provider's host). Thus, a user can simulate his complete system on geographically distant hosts. This functionality allows designers to avoid the licenses problems. In fact, distributed simulation does not require having the licenses and the simulators on the same machine. It permits distributing the simulation engines in the specific groups, generally having the licenses for the dedicated simulators. It is possible thanks to this method to combine the machines and the software in order to share available CPU resources for the various simulated subsystems, in order to speed up the whole system simulation.

Unfortunately, king of Peer-to-Peer IP based SoCs design and simulation tool don't exists yet. Its development constitutes the objective of our work.

This paper is organized as follow: in section 2, we give a review of state of the art on distributed simulation, IPs exchange and peer-to-peer tools. The concept of SoC P2P and use of our tool are explained over an example in the section 3. We conclude this paper in the section 4.

2. RELATED WORK

2.1 Peer-to-Peer Tools

A Peer-to-Peer tool allows users to communicate, collaborate or share files and applications without connecting to a web server. The connections between peers are direct (host-to-host), and data are not centralized on a server but distributed over the users hosts.

These tools are used more and more and applied to several domains as:

- Communication: ICQ, AIM
- File sharing: Napster, Gnutella (Gnutella), Freenet (Clarke, 2001), KaZaA, eDonkey
- Collaboration: Groove patform (Hurwicz, 2001), Microsoft NetMeeting
- Distributed computing: Netbatch ,SETI@home (SETI)
- Edge services : used for employee training in Intel (Spooner)

2.2 IP exchange

The last few years have been marked by the appearance of some tools allowing the IP exchange, in a secure way, between providers and consumers. The tools of

Design-and-Reuse (Design-and-Reuse) and Synchronicity (Synchronicity) are two significant examples. In fact, they permit to IP consumers to choice their component in providers IPs catalogs. Unfortunately, these tools deal only with the commercial aspect of IPs exchanging. Thus, they don't furnish tools assisting in the design or in the simulation. The data and the catalogs of these tools are centralized in an HTTP server.

2.3 Multi-level simulation

Recently many academic and industrial research teams worked on the multi level simulation problem in embedded systems. This type of simulation implies to execute together models of the system component described at different abstraction levels. Among the proposed solutions, the Bus Functional Model (BFM) (Semeria, 2002) constitutes the conventional methodology to interconnect functional simulation models and cycle accurate models, especially to validate software/hardware interfaces. Unfortunately this methodology takes into account only memory accesses, but it doesn't at all allow transformation of high level communication primitives (FIFO for example).

CoWareN2C (CoWareN2C, 2001) is an environment which offers a multi level cosimulation solution. It allows the use of two abstraction levels: BCA (Bus Cycle Accurate) which is closely similar to RTL and UT level (without timing references) Thus CoWareN2C presents a concept called BCASH, It is a wrapper of the sub-systems described at UT level allowing the estimation of their sub routines execution time. This wrapper can be automatically generated only in if the sub-system at UT level is targeted in software.(that means that the sub-system will be executed on a processor simulator ISS "Instruction Set Simulator"). This is unfortunately a very strong constraint.

Some very recent works, in the literature, treat the problem of automatic generation of multi level simulation models for heterogeneous multiprocessor SoC.

The work described in (Nicolescu, 2001) is one significant example. It permits to generate automatically simulation wrappers to adapt modules abstraction levels to the simulation level. Unfortunately, it doesn't target a specific class of application, and the wrappers are constructed by assembling basic components from external libraries. The structure of these simulation wrappers seems to be complex because of the important number of SystemC (SystemC), (Grotker, 2002) components (processes) present in each instance.

2.4 Distributed and multi-languages simulation

Sophocles is the European ITEA project dedicated to elaborate a distributed simulation platform. The solution studied in Sophocles (Boulet, 2003) is based on the where companies would not necessarily have to possess their simulation infrastructure and tools, but could rely on third-party companies whose specific business is to provide for a number of client companies a set of vcs (bought from vc providers) and a computing infrastructure capable of hosting part of simulations using these components.

We also find in the literature several tools allowing the geographically distributed simulation of SoC. For instance VCI and MCI (Hessel, 1998), (Hessel,

2000) developed in TIMA laboratory. In MCI the different modules of a SoC can be described in VHDL and in C language, the SoC can then be simulated on simulators residing in geographically distant hosts, by using RPC (Remote Procedure Calls) for the communication. VCI is a kind of extension of MCI. Indeed, it takes again the concept of the simulation geographically distributed with the same mode of communication (RPC), but it allows the use of more programming languages such as MATLAB.

Plug&Sim of Integrated Systems Inc. proposes a flexible simulation environment, where two simulators may run concurrently and communicate via a network using CORBA bus (ORB) (OMG, 2001).

Synthesia (Altmae, 2002) is a SW/HW co-simulation tool allowing simulating together: the hardware part of a SoC on a VHDL simulator and the software part as compiled C or ADA programs.

COSSAP (Runstadler, 1998) of Synopsys Inc. allows simulating simultaneously and concurrently modules described in C and assembling code with VHDL modules. It uses IPC for the communication between modules.

Some recent work (Nicolescu, 2001), (Yoo, 2001) describes a co-simulation environment based on wrappers assembling. They uses the Linux shared memory for the inter simulators communication. Thus, the co-simulation can be executed just on a single host.

In all these tools it seems that the distribution of the modules constituting a SoC on the different available hosts for the simulation is made manually and based only on the designer experience without any optimization.

These tools don't allow yet the distribution of SystemC specifications, and don't use the recent and systematic communication methods as CORBA.

2.5 Contribution

The main contribution of this work is to extend and to adapt the Peer-to-Peer concept to IP based SoC design and simulation. The communication will be realized using CORBA. This provides the tool a significant flexibility and heterogeneity.

3. SOC P2P

The architecture of our Peer-to-Peer tool pour IP based SoC design and simulation (SoC P2P) is similar to those used for the files sharing described in the section 2.

We will introduce the use of this tool over and example. Let's have an environment composed by four users (A, B, C and D). Each user has the SoC P2P tool installed in his machine, an IP library and the simulators able to simulate the available IPs in the local library. All the users are connected to Internet. This environment is illustrated in the figure 1.

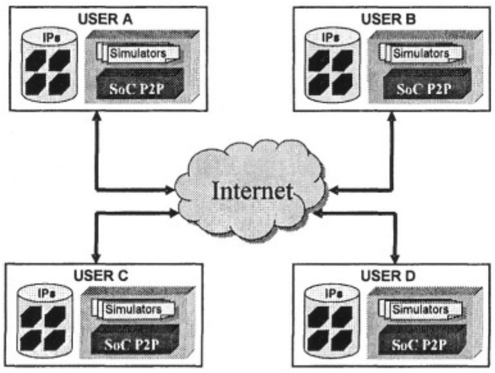

Figure 1 – A four SoC P2P users environment

The design project of USER D is given in the figure 2. The SoC to design is constituted by five IPs.

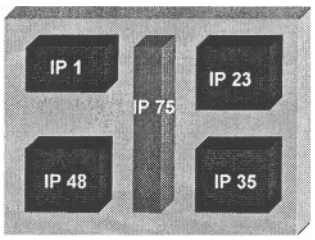

Figure 2 – Design project of the USER D

The IPs searching using SoC P2P tool gives the results shown in the table 1. Thus, the IPs 23 and 75 are available in the local library of the USER D. They are respectively described in SystemC and VHDL. The remaining IPs are available in foreign hosts of SoC P2P users. The users IDs of their owners and their descriptions languages are given by the tool. The last information given by the search operation is the simulation permissions. In fact, a SoC P2P user can allow or not any other user to use and to simulate the available IPs in his library.

Table 1- Results of IPs searching using SoC P2P

IP	Availability	Language	Simulation permission
IP 1	User B	SystemC	OK
IP 23	User D (local)	VHDL	OK
IP 35	User A	SystemC	OK
IP 48	User C	SpecC	OK
IP 75	User D (local)	VHDL	OK

3.1 Interfaces standardization

For more homogeneity and simplicity of the integration of the IPs in the distributed simulation models, we choose an emerging and very used communication standard called VCI. In fact, all IPs available in SoC P2P libraries have to be VCI compliant. This choice is motivated mainly by the two following reasons:

- Availability of well defined specifications of VCI interfaces and communication
- Simplicity of VCI communication
- Use of VCI standard in existing free IP libraries as SoCLib (SoCLib).

Thus, when a user has already an IP library not VCI compliant, simple interfaces can be written easily to encapsulate these IPs and make them VCI compliant.

3.2 Multi-level simulation models

Our simulation models generation flow generates a simulation module adapter for each module described at an abstraction level different from those on which we want to perform de simulation of the system. If we have for example a SoC composed by two modules: module 1 specified at UTF level and module 2 specified at RT level, and we want to simulate it at the RT level, our methodology generates automatically a simulation module adapter for the module 1 in order to adapt it's interface with those of module.

Our simulation module adapters are not constructed by assembling basic components for external libraries as in almost literature tools, but it is generated by rules composition. In our simulation models generation flow all simulation module adapters are instances of a SystemC class that we created and called "generic_interface".

The generic interface is a configurable SystemC class. It is composed, as all SystemC classes, by an interface file (.h) and an implementation/behavior file (.cc). The interface file defining the ports characteristics is automatically configured from the initial system specification, and the behavior file is constructed and configured by the composition of a set of rules. This step is described with details in (Meftali, 2003).

3.3 Communication

3.3.1 How does it work?

The principle of the communication in SoC P2P consists in using the CORBA naming service to locate a distant IP having just its name or its description. Then to encapsulate this IP, in a specific wrapper, to produce a "distributed IP" which can be used in a distributed SoC model.

This mechanism is completely transparent for the designer. Thus to design a SoC, if we want to use distant IPs, it will be enough to use some specific mechanisms described in the next paragraphs.

3.2 Creation of a distributed IP

The creation of a distributed IP, to be used in our simulation environment, do not change at all the current practices of the designers. In fact, any IP described in a system description language can be encapsulated in a SystemC module and become a distributed IP.

The method consists in encapsulating the IP (or a group of IPs) in a specific interface. This interface has the characteristic to be a SystemC module communicating with a CORBA object. This interface is called corba_server_if.

3.3 The transport Object

The transport object is, in fact, the CORBA object which will allow the invocation of distant methods. More precisely, it will make it possible to set and to get the values of the ports. It also allows synchronizing the distant simulator to which it belongs.

For each occurrence of this object, there is one SystemC simulator on the server side. This object is created by the server, and then the client can get an instance of it using the naming service. When the client gets any instance, it initialize it by calling it the method init().

The initialization of a transport object initializes also the SystemC simulator to which it belongs. The method doSim(…) permits to run simulation for one given period of time. This method is used to synchronize the client with the server. Indeed, it is the client who synchronizes the simulation, he calls the method doSim(…) to synchronize himself with server. The generation of the distribution wrappers is described with details in (Meftali, 2003 a).

4. CONCLUSION

In this paper we presented a tool called SoC P2P. It consists in adapting the Peer-to-Peer concept to the IP based SoC design and simulation. This is novel Peer-to-Peer application which can dramatically reduce the time-to-market in SoC design. The only restriction in SoC P2P is that the IPs must be VCI compliant. This is a standardization choice.

The distributed simulation models are automatically generated in a Multilanguage collaborative environment. The multi-level adapters are also generated in the case of abstraction levels heterogeneity in the design project.

5. REFERENCES

1. M. ALTMAE., Hardware/Software Coverification. Procs. of EDA Traff, March 1995.
2. T. GROTKER, S. LIAO, AL, System Design with SystemC. Kluwer Publishers. 2002.
3. PIERRE BOULET, JEAN-LUC DEKEYSER, CEDRIC DUMOUL1N, PHILIPPE MARQUET, PHILIPPE KAJFASZ, AND DOMINIQUE RAGOT. Sophocles: Cyber-enterprise for system-on-chip distributed simulation - model unification. In International Workshop on IP-Based SoC Design, Grenoble, France, nov 2003.

4. I. Clarke, T.W. Hong, "Freenet: A, Distributed Anonymous Information Storage and Retrieval in Designing Privacy Enhancing Technologies", Workshop on Design Issues in Anonymity and Unobservability, LNCS, New York,, 2001
5. COWARE INC.., Coware N2C. http://www.coware.com/cowareN2C.html, 2001.
6. DESIGN AND REUSE INC. http:// http://www.us.design-reuse.com/
7. eDonkey2000 home page, http://www.edonkey2000
8. Gnutella, www.gnu.org/philosophy/gnutella.fr.html
9. HESSEL F., LeMARREC Ph., VALDERRAMA C., ROMDHANI M., JERRAYA A. A., MCI - Multilanguage Distributed Co-Simulation Tool. DIPES'98, edited by F. Rammig, Kluwer. 1999.
10. HESSEL F., COSTE P., NICOLESCU G. all.., Multi-level communication synthesis of heterogeneous multilanguage specifications. (ICCD 2000), Texas, USA, September, 2000.
11. M. HURWICZ, "Emerging Technology: Groove Networks: Think Globally, Store Locally", Network Magazine, May, 2001
12. SAMY MEFTALI, JOEL VENNIN, JEAN-LUC DEKEYSER., Automatic Generation of, Geographically Distributed, SystemC Simulation Models for IP/SoC Design, 46th IEEE International MWSCAS, Cairo, Egypt, December 2003.
13. SAMY MEFTALI, JOEL VENNIN, JEAN-LUC DEKEYSER, A Fast SystemC Simulation Methodology for Multi-level IP/SoC Design, In International Workshop on IP-Based SoC Design, Grenoble, France, nov 2003.
14. Napster home page, http://www. Napster.com
15. NICOLESCU G., YOO S., JERRAYA A. A., Mixed-level cosimulation for fine gradual refinement of communication in SoC design. DATE'01, Munich, Germany, March, 2001.
16. OBJECT MANAGEMENT GROUP, (CORBA), Version 2.6. http://www.omg.org/technology/documents/formal/corba_iiop.htm, December, 2001.
17. OPEN SYSTEMC INITIATIVE, http://www.SystemC.org
18. P. RUNSTADLER and R. CREVIER.., Virtual prototyping for system design and verification. Technical report, Synopsys Inc., Mars 1995.
19. Search for Extraterrestrial Intelligence (SETI) home page, http://setiathome.ssl.berkeley.edu/
20. J. Spooner, K. Popovich, Intel: The future is peer
21. L. Semeria and A. Ghosh, "Methodology for Hardware/Software Co-verification in C/C++", Proc. of the ASPDAC, Jan. 2002.
22. SYNCHRONICITY INC. http://www.synchronicity.com/
23. K. Truelove, A. Chasin, "Morpheus Out of the Underworld", July 2001
24. YOO S., NICOLESCU G., JERRAYA A. A.., A generic wrapper architecture for multi-processor SoC cosimulation and design. CODES 2001, Copenhague, Denmark, April, 2001.
25. SoCLib home page, http://soclib.lip6.fr/