Reusing VHDL Soft-Cores by Means of Using Appropriate Workspace Management and Navigation Tools

Serafín Olcoz, José Luis Avellano, Lorenzo Ayuda, Ana Castellví, Isabel Hidalgo, Carlos Herrero
SIDSA. Ronda de Poniente 8, 2 A. 28760 Tres Cantos (Madrid). Spain
Pierre Plaza, Arnaud Morandeau, Juan Carlos Díaz, Jacobo Riesco
Telefónica I+D. Emilio Vargas 6. 28043 Madrid. Spain

Abstract

A VHDL Soft-Cores design reuse methodology has to be supported by a new wave of appropriate tools. Among these design reuse tools, this work presents the VHDL-ICE Workspace Management and its Navigation tools: The VHDL Design Unit Navigator, the VHDL Design Hierarchy Navigator, the Simulation Model Navigator and the VHDL Simulation Debugger.

The advantages offered by the VHDL-ICE environment and, specifically, by the navigation tools are presented from the tool developers' point of view, together with the user's perspective provided by a Spanish Telecom company when developing a circuit reusing Soft-Cores.

1 Introduction.

According to *VSI Alliance*[1] situation analysis, as semiconductor technology advances, the business pressure to design large ICs in a short time increases. Design reuse is expected to be a prevalent method for improving design efficiency of large ICs.

In many cases, the reused blocks are internally developed. However, even with the rapid advances in fabrication technology and design tools, few companies can dedicate to offer the customer a total "system-on-a-chip (SOC)" solution. Consequently, it is becoming critical for companies to increase their access to a variety of reusable functional blocks also called Cores, IP (Intellectual Property) or Virtual Component (VCs).

Reusable VCs are used in ASIC/IC design, embedded software design and board design. VCs can be three forms: Soft, Firm, or Hard. Soft VCs are delivered in the form of behavioral or synthesizable Hardware Description Language (HDL), e.g. VHDL, [1], and have the advantage of being more flexible and the disadvantage of not being as predictable in terms of performance. Soft VCs typically have increased IP protection risks because source code is required by the integrator. Firm VCs have been optimized in structure and in topology for performance through floorplanning and placement, possibly using a generic technology library. Hard VCs have been optimized for performance and mapped to a specific technology. Moreover, hard VCs depend on the availability of physical libraries for ASIC/IC design.

Reusing Soft-Cores implies a lot of engineering and reengineering support. Today, it's an engineer-to-engineer negotiation that requires technical experts on both sides of the negotiation, not just salespeople quoting prices and delivery. This technology-transfer process demands a design reuse methodology to be supported by a new wave of specific tools hosting such a methodology and the involved consulting services.

The proposed methodology takes advantage of software-like review and audit procedures supported by navigation tools[2], [2-3]. The paper presents these tools as part of the VHDL-ICE environment[3], [4], and the usage experience provided by a Spanish telecom company while developing a SOC design.

2 The VHDL-ICE: An Integrated Common Environment.

VHDL-ICE is a VHDL Integrated Common Environment, suitable for Soft-Cores design reuse. VHDL-ICE client/server architecture supports and interoperates across multiple platforms (Windows NT, UNIX), see figure 1. The client/server communication can be local, when both parts are integrated in the same executable implementation, or remote, when both parts are communicated by means of a middleware. This middleware is based on sockets and TCP/IP, allowing expandable and configurable installations over local or wide area networks (i. e., intranets and the Internet) without forcing VHDL developers to change their development tools or the way they work.

The VHDL-ICE database servers manage all derived design information required to control the different build processes of reusing a VHDL Soft-Core. Particularly those appearing during the simulation model processing stages as defined by the LRM (analysis, static elaboration and simulation).

[1] VSI Alliance stands for Virtual Socket Interface Alliance (http://www.vsi.org.). This alliance was formed from a common understanding of a looming bottleneck for the continued rapid evolution of the electronics industry and a shared vision for its solutions based on the continued use or "reuse" of existing functions, but which are designed in such a way as to make possible the mix-and-match of such functions from different sources onto a single silicon solution.

[2] These tools are part of the VHDL Simulation Assessment tools under development in OMI-ESPRIT IV Project REQUEST.

[3] VHDL-ICE is a leading VHDL Integrated Common Environment, for UNIX and Windows NT, under development in the OMI-ESPRIT IV Project TOMI.

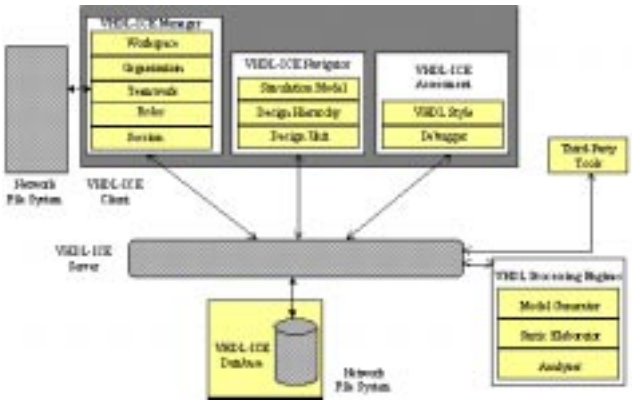

Figure 1.– The VHDL-ICE Organization.

The three main capabilities of the VHDL-ICE, ordered by implementation steps, are:

- **Session, Roles, Teamwork, Organization Units and Workspace Management**. This facility allows organizing multiple, flexible and reproducible workspace areas related to different organization units, such as VHDL Design Projects and Teamworks, [5]. The workspace of a given database is hierarchically distributed among organization units to which the users/teams belong to. Users have access to the different management tools of the VHDL-ICE Shell according to their particular roles and after opening a session on a given (local or remote) VHDL-ICE server.
- **Version and Configuration Management** facilities to ease VHDL design reuse, beyond the VHDL language's built-in facilities, during the development of a given design project as well as for the maintenance of the resulting IP. This capability together with licensing and protection issues will allow VHDL models become products and to be commercialized.
- Tools integrated in this environment, in a near future, will be part of a **Workflow and Design Process Control** facility. Therefore, VHDL-ICE will offer a design management facility that will allow fulfilling company procedures, such as coding styles, or satisfy ISO 9000 standards applied to VHDL design. The openness of this approach relies on the success of AIRE/CE and VHDL/PLI standardization processes, [6-7].

3 Role-Based Access Control, Teamwork, Organization Units and Session Management

A study of 28 organizations by the National Institute of Standards and Technology (NIST), [8], demonstrated that Role-Based Access Control (RBAC), [9], addresses many different needs in the commercial and government sectors.

Access policy is embodied in RBAC components such as role-permission, user-role, and role-role relationships. These components collectively determine whether a particular user is allowed access to a certain piece of system data. RBAC components can be configured directly by the VHDL-ICE Administrator or by appropriate roles as delegated by this administrator. Because the access control policy can, and usually does, change over the system life cycle, RBAC offers a key benefit through its ability to modify access control to meet changing organizational needs.

Although the RBAC concept is policy neutral, it directly supports three well-known security principles:

- Least privilege: Only those permissions required for the tasks performed by the user in the role are assigned to the role.
- Separation of duties: Invocation of mutually exclusive roles can be required to complete a sensitive task.
- Data abstraction: Instead of the read, write, execute permissions typically provided by the operating system, abstract permissions can be established.

On the other hand, RBAC cannot enforce the way these principles are applied. Theoretically, a VHDL-ICE administrator could configure RBAC to violate these principles. Also, the degree to which data abstraction is supported is determined by the implementation details.

Traditionally, specific applications have had to encode RBAC internally, with existing operating systems offering little environment or application-level RBAC support. VHDL-ICE offers a sophisticated RBAC including the capability of establishing relations between permissions and roles, and between users and roles. With RBAC, role-permission relationships can be predefined, which makes it simple to assign users to predefined roles, see figures 2 and 3.

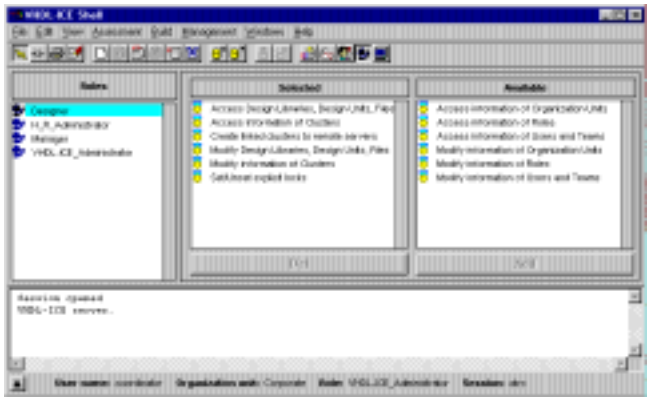

Figure 2.- The Roles Manager.

The NIST study indicated that permissions assigned to roles, unlike user membership in roles, tend to change relatively slowly. The study also found it desirable to let administrators confer and revoke user membership in existing roles without authorizing these administrators to create new roles or change role-permission assignments. One reason for this finding is that establishing the users and roles relationship requires less technical skill than assigning permissions to roles. Without RBAC, it can also be difficult to determine what permissions have been authorized for what users. The Teamwork manager

complements these features together with the possibility of grouping users into teams.

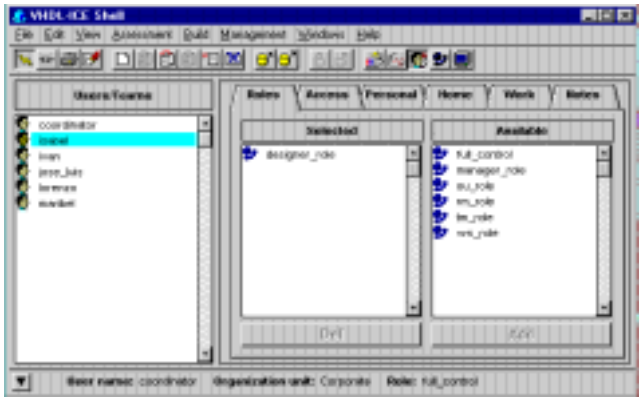

Figure 3.- The Teamwork Manager.

Users do not work alone or just as part of a group, but belonging to a given organization unit of the enterprise for which they use to work. An organization unit can be a company's division hierarchically composed of other organization units, such as department or projects, which, in turn, are hierarchically composed by other organization units, such as work-packages and/or tasks, and so on. Taking this reality into account, the Organization manager allows to map users with their corresponding roles to a given organization unit at any level of the hierarchy of organizations of a given database, see figure 4.

This manager also distributes parts of the whole workspace of the selected database to the organization units in a hierarchical manner. Users belonging to a given organization unit could have access to their workspace with one of the selected roles when opening a VHDL-ICE session. The Organization manager also allows to select the set of VHDL Style rules that can be checked to any VHDL design available in the workspace of a given organization unit during a session.

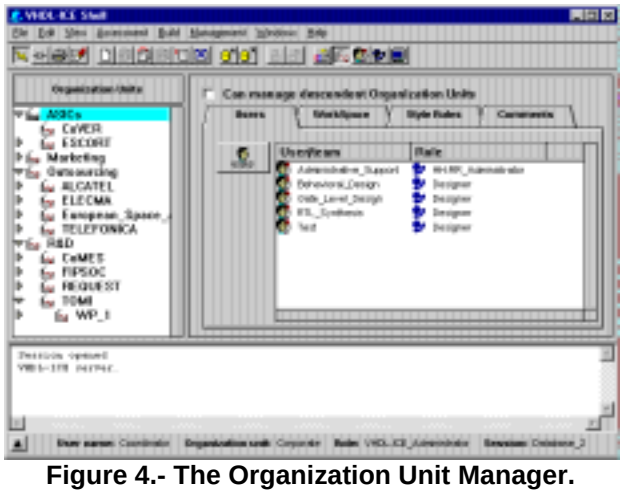

Figure 4.- The Organization Unit Manager.

VHDL-ICE users establish sessions during which they may activate a subset of the roles they belong to. Each session maps one user to possibly many roles in many organization units to finally be able to reach the corresponding workspace, if any. The concept of a session equates to the traditional notion of subject in the access control literature.

A user might have multiple sessions open simultaneously, each in a different window on a workstation screen, corresponding to a different VHDL-ICE Shell or client, see figure 5.

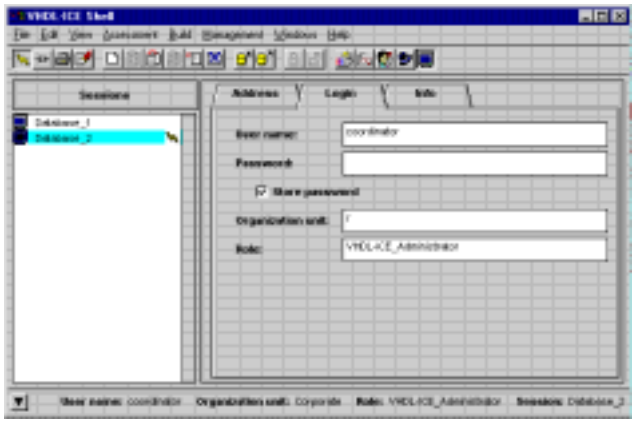

Figure 5.- The Session Manager.

These VHDL-ICE management tools are also the Graphical User Interface (GUI) of the VHDL-ICE database metadata that is not directly associated with the design data, but with the organizational and administrative aspects. These design project management aspects, while they are very relevant to their own success, are not usually integrated, or even considered at all, into the state-of-the-art Electronic System Design Automation (ESDA) processes.

Thanks to these management tools, the VHDL-ICE is a unique and powerful framework for setting up a shared and distributed workspace where a design reuse methodology can be more easily put in place than in a stand alone environment based on reusing VHDL design files with just the operating system support.

The remaining VHDL-ICE management tool is the Workspace one. This tool is the real core of the VHDL-ICE design management navigation environment. The workspace management tool is also the GUI of the VHDL-ICE database metadata that is directly associated with the design data, i.e., the VHDL design objects and the associated information, e.g. simulation and synthesis scripts, design and style errors information and so on.

4 Workspace Management and Navigation Tools.

The ability to access information and act on it quickly will become increasingly critical to any company's (or individual's) design reuse success. Raw data becomes information when it gets into the hands of someone who can put it in context and use it. The data is the raw ingredient, which makes all this possible. There are many parallels between the manufacturing and distribution of discrete components and the distribution of information related to Soft-Cores. Making decisions using old, incomplete, inconsistent, or invalid data puts a Soft-Cores design reuse methodology at a disadvantage

versus the competition. Moreover, when an organization is able to examine Soft-Cores information over a lengthy period of time, design reuse trends and patterns become apparent that simply are not observable in current Soft-Core information by itself. The Workspace management and the associated navigation tools of the VHDL-ICE environment provide the appropriate tool support for an adequate implementation of a successful design reuse methodology. Such a methodology unavoidably implies to apply reverse engineering and reengineering techniques on the available information that is better managed by the VHDL-ICE database server than by the file system of any operating system.

4.1 Workspace Management.

This tool allows to directly explore and deal with the primary processing units of any VHDL model (the VHDL Design Units), the primary hardware abstraction of VHDL (the VHDL Design Entities), their dependencies (clients and suppliers), their corresponding design and style errors and associated information, such as simulation and synthesis scripts.

The Workspace management tool has a very useful import/export facility to open a communication channel between the objects controlled by the this environment and the corresponding files available in the file system that are used by the existing design flows (i.e., simulation, synthesis, test and other tools) of the users without any interference imposed by the usage of the VHDL-ICE environment. This feature means the possibility of reproducing the hierarchical structure of the information available in the workspace in a given hierarchical structure of files in the file system, and viceversa.

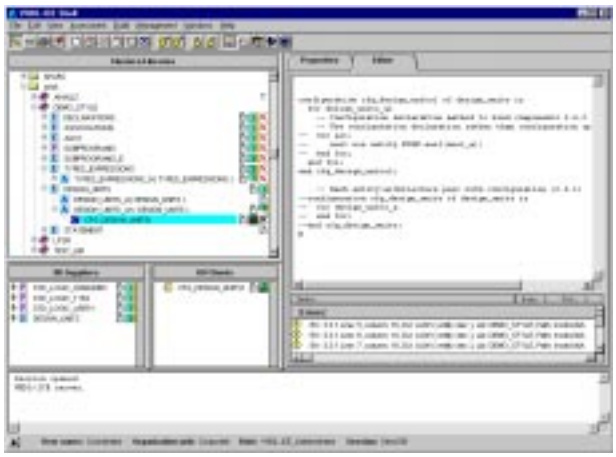

Figure 6.- The Workspace Manager.

It is possible to control the build products and the VHDL build process used to produce the VHDL soft-Cores based on a given configuration independently of where their components are located in the distributed workspace over several databases available in different platforms, their current build processing and dependencies state. When VHDL design or style errors are identified in one of the components or in one of their clients or suppliers, the integrated editor helps to identify and fix them.

The Workspace management tool allows to examine and navigate through the (local or remote) clusters in which it is split, the contained VHDL Design Libraries or its corresponding contained VHDL Design Units and associated information. When focussing on one of these objects, information about its properties is shown and, if it is an editable object, then the contents is also available and refreshed according to the targeted object selected in the workspace, see figure 6.

Besides these navigation facilities outside of a given VHDL Design Unit offered by the Workspace manager, it is also possible to navigate and examine the information inside of them. This is done by means of the complementary navigation tools, presented in the next subsections, that will be completely integrated into the Workspace management tool in the near future.

4.2 VHDL Design Unit Navigation.

The VHDL Design Unit Navigator works inside of VHDL Design Entity Declarations, Architecture Bodies, Package Declarations and Package Bodies. Its navigation capabilities offer a very good support, complementary to the VHDL source code edition, to perform Soft-Cores HDL description inspection and walkthrough processes. The usage of this tool is particularly indicated while developing a new Soft-Core and when acquiring a Soft-Core that is not very well documented or their in-line comments are not enough to understand the functionality or the usage of the recently acquired IP, see figure 7.

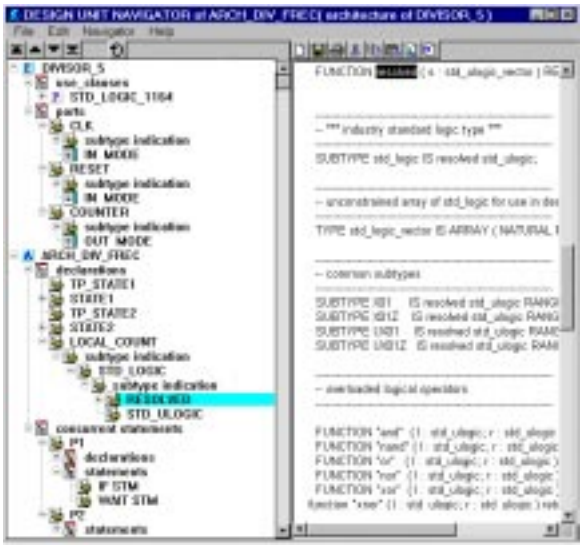

Figure 7.- The VHDL Design Unit Navigator.

This tool also includes interactive edition capabilities. Now, using the tool, it is possible to edit a new VHDL Design Unit, or an existing one, and to navigate through all the references that are being used or needed to use and there is not a well knowledge about them or how to use them. For example, when referencing

a subprogram and the definition of the parameters and even the location of the subprogram is unknown and requires a lot of source code navigation through the library to have access to this information. Another example, is to quickly and automatically find the declaration, with all the related information, of a given VHDL design objet that is needed to use or to understand.

This navigator is the closest related tool to those probably used for supporting software inspection code and walkthrough processes, [3]. The other navigators are more related to the compositional and hierarchical nature of the hardware described in VHDL and the own simulation semantics of the language, respectively.

4.3 VHDL Design Hierarchy Navigation.

The VHDL Design Hierarchy Navigator deals with the hierarchical structure or components' netlist of a given VHDL Design Entity, see Figure 8. This navigation tool also informs about connectivity and the places where a given component is used.

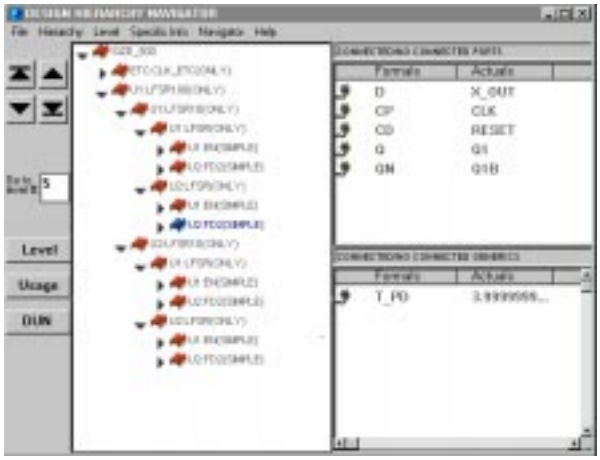

Figure 8.- VHDL Design Hierarchy Navigator.

The tool can translate a components based VHDL description into a block statements based one or into a flatten process statements based one. These capabilities are useful to automatically support specific corporate or coding style guidelines without reshaping by hand a given external source code to be policy-compliant. Following this direction, i. e., to enhance reusability, readability and maintainability of the Soft-Cores, this navigation tool automatically generates a configuration declaration from a given VHDL Design Hierarchy of a given Soft Core, at any chosen level of the hierarchy.

This navigator is useful for supporting both review processes of Soft-Cores: HDL description inspection and walkthrough. Its power to this end is increased when combined with the favorite text editor and other navigators. On the other hand, it is particularly indicated to analyze descriptions automatically generated by a tool, e. g. a synthesis tool, or by a foreign design team when reusing IPs.

The tool is being enhanced with VCs edition capabilities to allow changing the design hierarchy configuration on the fly, without changing first the source code by hand. These capabilities will also allow to change the configuration, and/or the ports and generics mapping on the fly. The changes in the corresponding source code to be produced as a result of these edition capabilities will be done automatically and transparently for the user. These capabilities will allow to reuse library components as if the user were directly dealing with the VCs.

4.4 VHDL Simulation Model Navigation.

This navigator allows to navigate through the underlying simulatable model (the heterarchy of VHDL processes interconnected by nets), see Figure 9. To keep the relationship with the corresponding design hierarchy and design unit information, this tool can be used in combination with the VHDL Design Unit and the Design Hierarchy Navigator, respectively.

Although this navigation tool is also very useful for HDL inspection and walkthrough, its scope is related with the simulation semantics underlying to any VHDL description, [10], and, for example, there is no way to deal with aspects related to the synthesis or hardware semantics of the corresponding VHDL description. The other two navigators are not constrained by this semantics and the result of their use for statically analyze a Soft-Core is totally independent of the further usage to which the VHDL description is oriented (e. g., synthesis purposes).

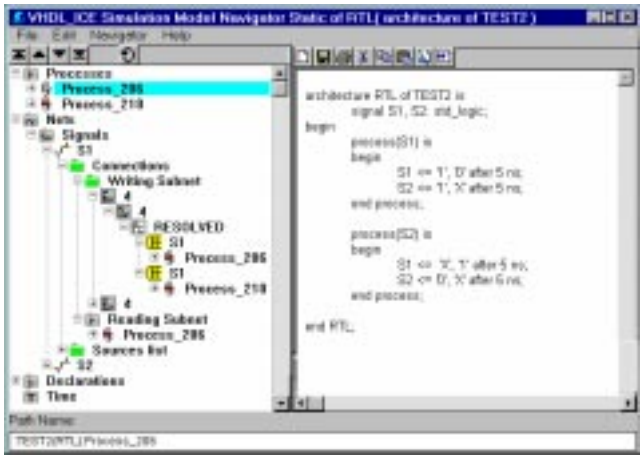

Figure 9.- VHDL Simulation Model Navigator.

The most powerful capability of this tool is to show information about the nets of the simulatable model that it is normally internal information only managed the VHDL simulation tools. This information helps to better understand which components, processes or statements are writing a given signal and how it is transformed before providing the corresponding driving value, and which components, processes or statements are reading or waiting for a given signal.

Finally, it is also possible to edit the source code of the corresponding VHDL Design Unit of the chosen object, in the simulatable model, and to automatically update this model to continue the navigation and/or edition tasks.

4.5 VHDL Simulation Debugger.

The VHDL Simulation Debugger allows to perform a dynamic analysis of the simulatable model which is complementary to the one performed by the VHDL Simulation Model Navigator. The evolution of the simulatable model, corresponding to a given VHDL Design Entity, with respect to the time is usually known as a VHDL simulation.

The VHDL Simulation Debugger is a navigation tool that allows to examine the state of a simulatable model at any given time. This tool allows to analyze this state with respect to the source code, making references to the related Design Units to which it depends on, the design hierarchy and the network of processes and signals.

The amount of information managed by this tool makes of it a low performance simulator. Because of this, it is better classified as a debugger with powerful analysis and navigation facilities.

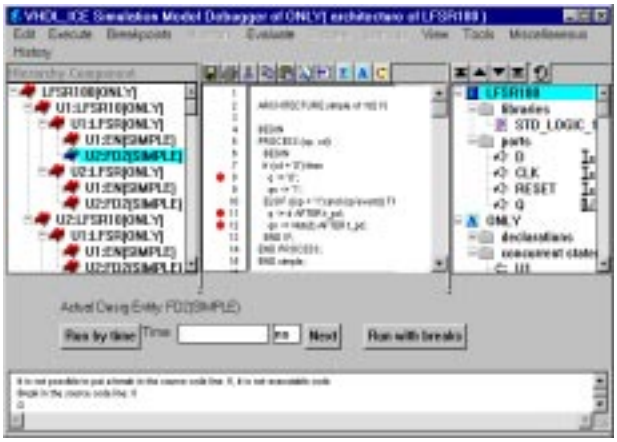

Figure 10.- VHDL Simulation Debugger.

This navigation tool closes the cycle of exploiting the VHDL intermediate formats produced during all VHDL compilation stages (analysis, elaboration and simulation), [10]. All together, combined with the error editing capabilities make of VHDL-ICE environment the appropriate navigation tools for supporting VHDL Soft-Cores reuse.

5 Usage Experience.

The Hardware Technology Division of Telefónica I+D (TID) has a wide experience in IC design. It has been pioneer in circuit design using HDL's and Synthesis tools, and this design philosophy is now deeply established. TID, as part of TOMI project, is participating in the VHDL-ICE demonstration by designing a circuit using this environment, giving feedback to SIDSA's development team, from the user's point of view, and allowing its test and validation with a real circuit example.

5.1 User Application.

As Research and Development branch of Telefónica, the Spanish Telephone Company, TID is very involved in the telephone business and its future possibilities. TID designed some years ago the Telefónica pay phone system, which is now deployed not only in Spain, but also in many other countries in South America. This system, based on a 8 bits microcontroller, is now under revision and updating, and as prevision for the future system, TID is now designing a microcontroller specifically oriented to pay phones. The special characteristics of the pay phone system impose some design restrictions:

- The new system has to be very modular, and the microcontroller must include the complete pay phone system functionality, so specific controllers are needed: phone keyboard, LCD Display, communications (USART's, I2C interface...).
- The power consumption allowed to the complete system is very small (it is battery supplied), so the microcontroller has to be very sparing at this point.
- Apart from a very low power consumption in active mode, it needs to have the possibility of "going to sleep" in several low consumption modes.
- In order to have a reduced design time, the microcontroller has to be based on a standard system, using IP modules and architectures.
- The future needs have to be taken into account; the microcontroller must be ready to include RTOS, Web Technology, remote operation, etc.

The MicroPP (Microcontroller for Pay Phones) is a 32 bits microcontroller based on ARM7TDMI, using Advanced Microcontroller Bus Architecture (AMBA). This application fits the objective of being a good demonstrator for the TOMI project:

- It is a quite complex telecommunications circuit (around 60 thousand equivalent gates).
- It includes Hard Cells (ARM7TDMI, USART module from ATMEL, I2C module from PHILIPS), Soft-Cores (AMBA modules from ARM), and brand new modules (now under development in TID).
- The design task is divided between electronic designers, who are co-operating in the complete design, but they are individually in charge of different circuit portions.

5.2 Demonstrator Circuit Design Flow.

The design flow based on Hardware Description Languages (HDL) and Synthesis has become the standard in Microelectronics. The Design of MicroPP circuit is being developed in VHDL, taking advantage of the previous experience in this field acquired in TID.

The traditional HDL design flow starts with the circuit specification, and the designer's task is to interpret the specifications and convert it into a synthesisable HDL model. The hardware designer is the one and only responsible of the model quality and fulfillment of the requirements. Once the model is considered good enough, it is synthesised using automatic tools. Finally, the result from the synthesis is used as starting point in the circuit place and route. During the complete design phase, the designer is also responsible of testing the circuit functionality.

With the usage of IP cores the designer is released from a lot of burden work, but he is now forced to check modules he did not designed, even without knowing the way they work. Then the designer needs some help from the tools for having measures about the quality and state of the IP cores. Sometimes, the functionality of the IP modules has to be changed to fit the requirements of the new application, and then an accurate version management is required. In any complex design, with many design modules from different sources and in different design stages (behavioral, RTL, gates), it is required a design management as friendly as possible.

The step between specification/modeling and synthesis has now been enlarged, including many new tasks the designer has to accomplish. It is at this stage where the new tools like the VHDL-ICE navigation tools must provide support to reduce the design time and improve the quality of the work.

5.3 VHDL-ICE Benefits.

The design tasks of the demonstrator circuit are being eased by the usage of the VHDL-ICE environment. Particularly, this environment, and its navigation tools, cover the hole between the circuit specification/modeling and the further synthesis, where the designer had to work alone before.

The environment is useful for design teams, taking into account each designer works with different modules, but these modules are interconnected and used by other designers. The RBAC access control and the version management co-ordinate in a natural way the work done by different people, and avoid the usual inconsistency problems.

The navigation tools provide a powerful way to analyze and debug both, IP cores and user designed modules, and the VHDL Style tool allows a good measure of their code quality.

Finally, the design management is independent on the design stage, and the design structure can easily be exported and imported back in different design phases (behavioral models, RTL models, netlists pre and post layout...).

6 Conclusions.

The RBAC system of the VHDL-ICE environment to share VHDL Soft-Cores libraries located in a distributed network of databases provides a powerful design reuse framework. The VHDL-ICE Workspace Management and the Navigation tools have been presented as appropriate tools for supporting a VHDL Soft-Cores design reuse methodology. Particularly for understanding the hidden know-how in Soft-Cores developed by third parties that have to be reused without the availability of the developers and/or the appropriate documentation.

The users' experience has also shown the need for applying some kind of reverse engineering or reengineering tasks when reusing VHDL Soft-Cores in the development of a SOC. These tasks have been carried out in a parallel manner, showing the need for supporting multiple and shared access in a control way as the one provided by the VHDL-ICE environment.

Similar tools to the ones presented in this work will presumably appear very soon as complementary tools to the already existing design tools such as VHDL simulators, synthesis tools and so on. This new wave of tools will better support the reuse of VHDL Soft-Cores and by that time the navigation tools and the environment presented here will be considered as pioneering tools for the new and exciting Soft-Cores design reuse methodology and, even more important, Soft-Cores reuse business.

References.

[1] "1076-93 IEEE Standard VHDL Language Reference Manual", IEEE Inc., New York, N.Y., U.S.A., September 1993.

[2] S. Olcoz, A. Castellví, M. García ."Static Analysis Tools for Soft-Core Reviews and Audits", Design Automation and Test In Europe Conference (DATE'98). Paris, Francia, February 1998, pp. 935-936.

[3] S. Olcoz, A. Castellví, M. García . "Improving VHDL Soft-Cores with Software-like Reviews and Audits Procedures", VHDL International Users' Forum, Spring Conference. Santa Clara, CA (VIUF'98), March 1998, pp. 143-146.

[4] S. Olcoz, L. Ayuda, A. Castellví, M. García, I. Izaguirre, O. Peñalba, "Implementing a VHDL Design Manager: VHDL-ICE." VHDL International Users' Forum, Spring Conference. Santa Clara, California, April 1997, pp. 93-102.

[5] S. Olcoz, L. Ayuda, I. Izaguirre, O. Peñalba "VHDL Teamwork, Organization and Workspace Management", Design Automation and Test In Europe Conference (DATE'98). Paris, Francia, February 1998, pp. 297-302.

[6] J. Willis, R. Newshutz, P. Wilsey, D. Martin, G. Peterson, J. Hines, A. Zamfirescu, "Advanced Intermediate Representations with Extensibility (AIRE)", VHDL International Users' Forum, Fall Conference, pp. 33-42, October 1996.

[7] F. Martinolle, D. Corlette, S. Pattanam, "A VHDL Procedural Interface for VHDL: VHPI", IEEE/VIUF International Workshop on Behavioral Modeling and Simulation, Washington, October 1997.

[8] D. F. Ferraiolo, D. M. Gilbert, N. Lynch, "An Examiniation of Federal and Commercial Access Control Policy Needs," Proceedings NIST-NCSC National Computer Security Conference, 1993, National Institute of Standards and Technology, Gaithersburg, Md., pp. 107-116.

[9] R. S. Sandhu, E. J. Coyne, "Role-Based Access Control Models," IEEE Computer, February 1996, pp. 38-47.

[10] S. Olcoz and J. M. Colom, "The Discrete Event Simulation Semantics of VHDL", in Proceedings of the International Conference on Simulation and HDLs, San Diego, CA, 1994, pp. 128-134.