VHDL Teamwork, Organization Units and Workspace Management

Serafín Olcoz, Lorenzo Ayuda, Iván Izaguirre, Olga Peñalba
SIDSA
Ronda de Poniente 8, 2 A.
28760 Tres Cantos (Madrid) Spain
vhdl-ice@sidsa.es

Abstract.

A new set of tools for Teamwork, Organization Units, Workspace and Build management of VHDL-based reusable components, organized in libraries, accessible through an heterogeneous and distributed environment is presented. These tools support the collaborative and distributed development of systems-on-a-chip reusing VHDL components available through the intranets and the Internet. They must be used as a complementary support to the design tools (simulation, synthesis, etc.) already available in the market to enhance productivity, facilitating maintenance, improving reliability, efficiency and interoperability, and finally, capitalizing on the IP library components investment.

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" solution. Consequently, it is becoming critical for companies to increase their access to a variety of functional blocks (often referred to as IP, or Intellectual Property).

Addressing this point, a library based approach and design reuse of hardware components is presented in section 2 of this paper. However, the availability and management of reusable Hardware Description Language (HDL) based libraries is very dependent on the particular processing mechanisms of the chosen HDL, e. g. VHDL, [1]. Section 3 presents the Teamwork, Organization Units, Workspace and Build management tools. These tools are part of a VHDL Integrated Common Environment (VHDL-ICE[2], [2]) and they have been developed taking care and advantage of the specific processing and interoperability mechanisms of VHDL descriptions, [3]. Finally, section 4 presents the conclusions and future work.

2. Library-Based Approach and Design Reuse.

Technology advances should be encapsulated into modular building blocks supported by new system design tools that enable the system designer to effectively combine and reuse them and evaluate the best solution. These modular blocks can be considered as the reusable elements of a design library. Making these library components reusable holds a number of promises, such as enhancing productivity, facilitating maintenance, improving reliability, efficiency and interoperability, and finally, capitalizing on the library components investment.

The reusability of the library components has not the same meaning depending on your point of view as IP user or as producer. The first five promises are useful for the (re)user of the components. However the last two promises are more related with the library components producer. On the other hand, modeling rules and coding styles increases the readability and modifiability of the library components.

[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] VHDL-ICE is a leading VHDL Integrated Common Environment, for UNIX and Windows NT, under development in the OMI-ESPRIT Project TOMI. See http://www.omimo.be.

Dealing with libraries means to enhance the library building and management facilities. The building facilities mean the capability of building systems from the library components. The management facilities mean the capabilities to organize these components. Regarding both criteria, no library component should be accepted into the library unless it meets a set of quality criteria, defined precisely by the company as part of its reusability policy. To help the company in this job, some quality checker must be available.

The goal of a reusability policy is to satisfy the users/customers (the potential re-users of the library components) by prevailing on the producers (the library components designers) to do the best possible job. However, no library component is reusable unless it is useful and usable, which is not the same thing.

It is important to state that reusability can not be added as an afterthought and that no library component is really reusable until it has been already reused. For this reason, it is possible to establish 5 levels of library component reusability, see figure 1:

1. Used successfully in one system.
2. Used in several systems produced by the library component's provider.
3. Used in systems provided by the designer's team.
4. Used in systems produced by third parties, all of whom the library components' designer knows about.
5. Used in systems provided by designers of whom the library components' author has never heard.

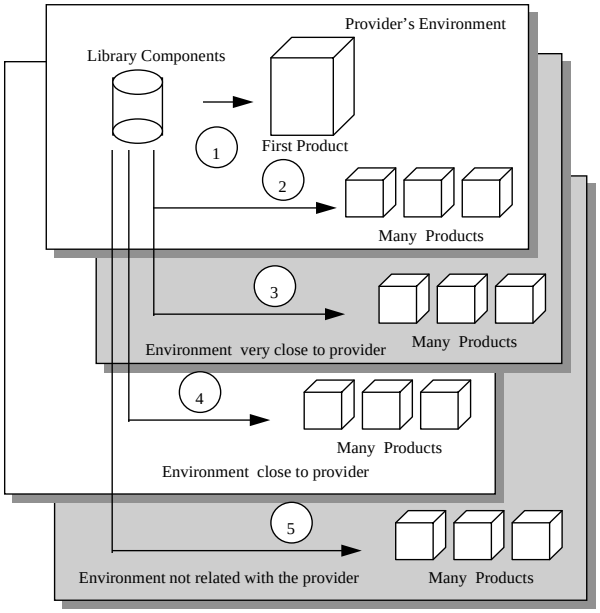

Figure 1.- Reusability Levels.

Each progression to a new level on this list brings a new set of requirements since it may reveal hidden assumptions on the environment. Moving to level 2 means removing dependencies on the original application of the library components. At level 3, the people reusing the library components are working at the same company (division), or at least are close related to the library components provider, but they are not the provider, so some of the components provider may turn out not be valid any more. Moving to level 4 means to start delivering reusable library components to people outside of the company (division), and that may cause some surprises. But at this level the relationship between the IP customers and the provider can solve the possible problems. At the highest level, this is not true any more: people know about the library components even the provider does not know about them.

3. VHDL-ICE: A Distributed and Heterogeneous Environment.

To use and reuse the library components not only the hardware/software modules are needed, but also an environment for its reuse. While the benefits and advantages of such a kind of environment are widely agreed, there appears to be very little commercial use of them that are currently available. Particularly, HDL-based hardware development is mostly supported by means of customization of software-like development oriented environments, [4]. In this approach the HDL data and operations management is carried out using the typical mechanisms to configure software development projects without taking into account that the HDL description is just a mean to deal with a hardware product and the software programs are themselves the product. This approach does not take advantages of the semantics behind the HDL. A different and innovative approach is the one followed by the VHDL Design Management tool called VHDL-ICE.

The VHDL-ICE environment relies on an object oriented distributed system architecture[3] implemented over TCP/IP, allowing expandable and configurable installations over local or wide area networks, i. e., intranets and the Internet. Users of the VHDL-ICE access to a particular environment, independent from the local machine used, offered by a chosen VHDL-ICE database server. VHDL-ICE client/server architecture supports and interoperates across multiple platforms (Windows NT, UNIX). VHDL-ICE data and metadata constitute a repository or database based on the standard file system.

The VHDL-ICE database server is characterized as a transaction processing system, which performs transactions

[3] SIDSA's ESDA (Electronic System Design Automation) tools are implemented in Eiffel, [5], using Integrated Software Engineering Inc. (see http://www.eiffel.com) tools, and ANSI C languages.

on the design data to take it from one consistent state to the next consistent state. During a working session, a user may access data over a distributed set of VHDL-ICE databases on a transparent way.

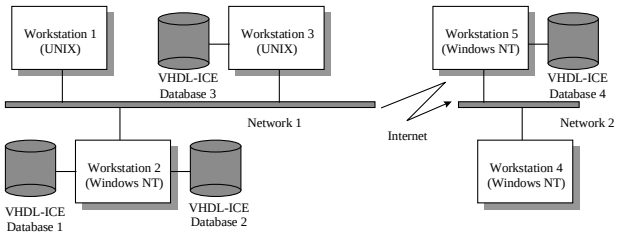

Figure 2.- Distributed Implementation of the VHDL-ICE Environment.

Figure 2 shows an example of possible distributed configurations over a network, [6].

3.1.The Main Capabilities of VHDL-ICE

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

1. **Workspace Management**. This facility allows to organize multiple, flexible, and reproducible workspace areas related to different organization units, such as Design Projects and Teamworks. The workspace management also includes VHDL build management capabilities that can be considered as the "built-in" VHDL configuration management
2. **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.
3. VHDL tools integrated in the VHDL-ICE 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 to fulfill company procedures, such as coding styles, or satisfy ISO 9000 standards applied to VHDL design. The openness of this approach relies on the availability of AIRE/CE and VHPI standardization processes, [7-8].

3.2. VHDL-ICE Teamwork Management.

This management tool allows to administer the VHDL-ICE users (designers and managers) and teams (groups of users) to whom the access to a given VHDL-ICE database server is permitted. A VHDL-ICE server administrator manages the information recorded about them, see figure 3.

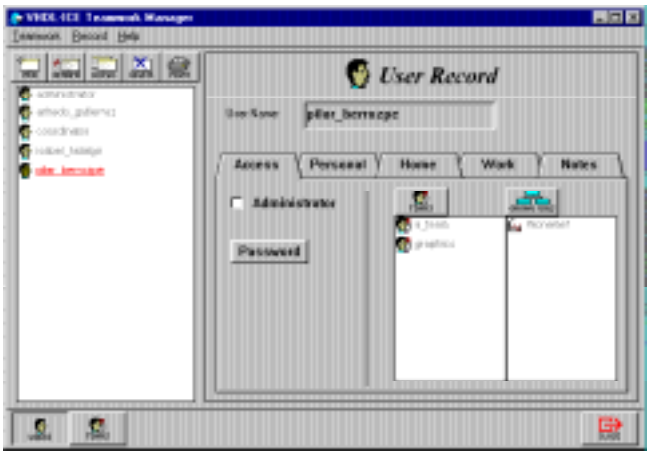

Figure 3.- VHDL-ICE Teamwork Management.

There is, at least ,one administrator per VHDL-ICE database server available trough the network.

3.3. VHDL-ICE Organization Units Management.

Any organization can be described as a hierarchy of organization units, e. g. a company can be considered as composed by departments, which are composed by projects, composed by tasks, and so on. The granularity of the chosen organization units allows to manage a company, department, project, etc.

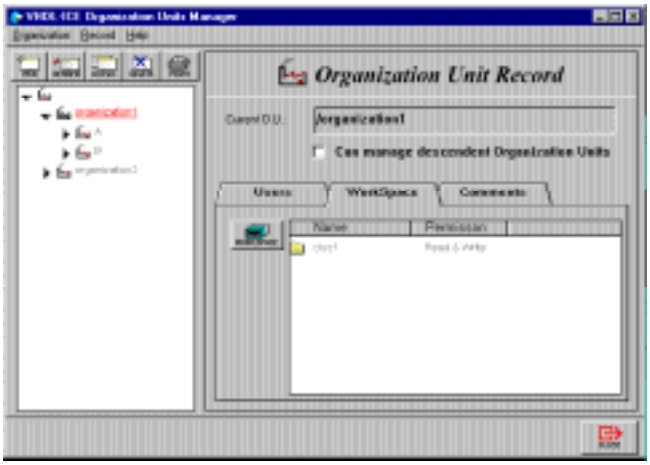

Figure 4.- VHDL-ICE Organization Units Management.

The Organization Units manager allows to associate to a given organization unit the corresponding available resources, see figure 4. Any user opening a session has to chose a working organization unit among the ones to which she belongs. Then, she has access to the available resources of the chosen organization unit, e.g., the corresponding workspace with the given privileges. In a near future, the resources will also include the available design tools (for simulation, synthesis, and so on) and IPs licenses.

3.4. VHDL-ICE Workspace and VHDL Build Management.

Software-like build management is based on dealing with source code dependencies analysis by means of home-grown build scripts and the *make* program, [9-10]. In this approach, the finest management granularity level are the text files of the File System (FS), where the source code is stored. State-of-the-art VHDL design tools are trying to apply this build management approach to VHDL-based designs. However, VHDL build management can not be properly addressed by such approach as it will be shown in this section.

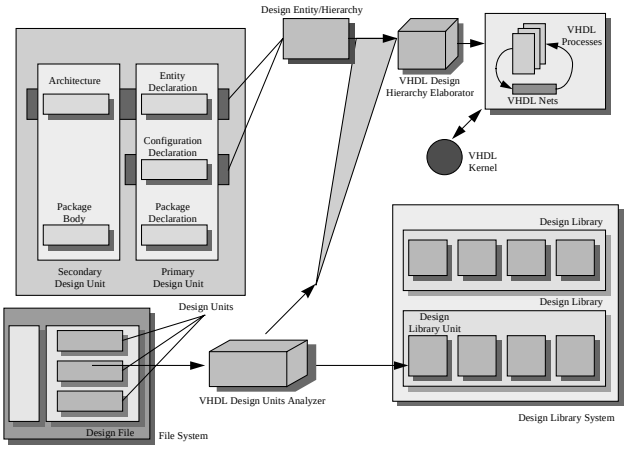

Figure 5.- VHDL Design Units Analysis and Design Hierarchy Elaboration.

According to the VHDL Language Reference Manual (LRM), [1], VHDL source code is organized in Design Units (DUs) that may be independently analyzed and inserted into a Design Library (DL). The DUs are sequentially stored and analyzed in text files called Design Files (DFs), i.e., implementation-dependent storage facilities for previously analyzed DUs. A DU is composed by a context clause and a library unit (Primaries: *Entity Declaration*, *Configuration Declaration* and *Package Declaration*; and Secondaries: *Architecture Body*, *Package Body*). The context clause defines the environment in which a DU is analyzed to check its syntax and static semantics conformance with the LRM, [3]. This environment is composed by the library units referenced within the DU being analyzed. The analysis of a DU defines the corresponding library unit in the DL. It seems to be that the finest granularity relevant to VHDL build management is not the DF but the DU.

On the other hand, the primary hardware abstraction in VHDL is not the DU, but the Design Entity (DE), which is defined by two DUs, an Entity Declaration together with a corresponding Architecture Body, and can be optionally configured by a Configuration Declaration. To identify all DUs involved in the composition of a given DE it is needed to statically elaborate the corresponding Design Hierarchy (DH), see figure 5. So, even constraining the number of DUs available in a DF just to only one, i.e., forcing to identify a DF with a DU, will not be enough to offer the possibility of properly using software-like build management techniques to deal with DEs. Therefore, VHDL build management has to be performed at DU and DE levels and a work place for dealing with these VHDL design objects has to be developed.

The VHDL-ICE workspace is a hierarchical collection of clusters containing other clusters and/or DLs. The content of a given DL can be shown by means of DUs or DEs. Therefore, the Workspace Management tool deals with the two kind of VHDL design objects relevant to VHDL build management, i.e., DUs and DEs, see figure 6. However, this tool manages much more information. In fact, the VHDL-ICE data are all the possible processing or building stages of a given DU or a DE, i.e., the DU's source code, the Abstract Syntax Tree (AST) resulting of the DU's parsing (the VHDL AST, VAST), the AST resulting of the DU's analysis (the VHDL Intermediate Format, VIF), the AST resulting of the DE's static elaboration (the Elaborated AST, EAST) and the AST resulting of the simulation model generation of a given DE (the Simulatable AST, SAST), [2]. The VHDL-ICE data can be shared among users/organization with different access privileges (read, write and IP in the near future) to avoid unnecessary rebuilds of shared DUs and/or DEs. So, the Workspace Manager implicitly includes all required VHDL build management facilities.

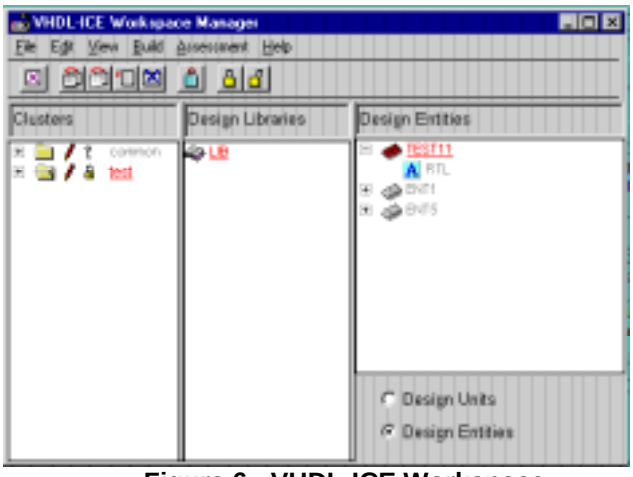

Figure 6.- VHDL-ICE Workspace Management.

The VHDL-ICE Workspace Management tool offers the common edit options to manage the workspace objects: create, delete, copy, move and rename. It allows the introduction of already edited VHDL descriptions into the

VHDL-ICE environment by importing the DUs of a given set of DFs and placing them in the chosen DLs. It is also possible to extract information from the VHDL-ICE environment by exporting a piece of the current workspace data into the FS.

The tool also provides relevant and very useful information about the workspace objects not directly related with their building mechanisms, e. g. it could be information for IP commercialization about compliance with the industry standards or VSI, market segment, company, contact point, system level application notes, datasheet, synthesis scripts information and so on.

The tool sustains the collaborative work, it lets that multiple users share a common repository through their workspaces. To deal with the shared access to data, the VHDL-ICE implements three locking layers: a general locking system to avoid concurrent modification of an object; a lock defined in the LRM, needed by the nature of the objects engaged, that is set on the resources during the analysis; and a persistent locking system to provide an enhanced management control on the shared data. While the first two lock kinds are automatically applied by the VHDL-ICE, the last is an additional tool for the managers and designers used to forbid the concurrent modification of objects or to ensure an invariant state.

The Workspace Management tool gives updated information related to the collaborative work, i.e.: how objects are shared, which are the access rights during a particular session, or who made changes to a particular design object. From this tool, data from distributed repositories can be managed. This tool can request connections with multiple VHDL-ICE servers to deal, in an integrated way, with distributed design objects. This allows an easy manipulation of distributed data as design objects in heterogeneous systems.

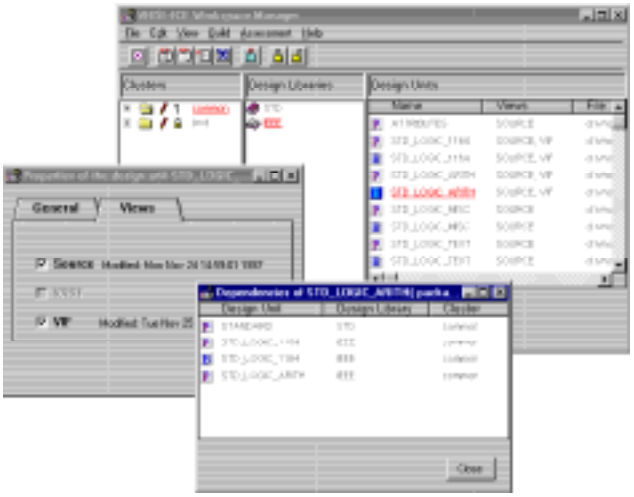

Figure 7.- DU's Closure.

The build management capabilities of VHDL-ICE properly deal with DUs and DEs by means of including VHDL analysis and static elaboration processing steps to take advantage of the built-in configuration mechanisms provided by the VHDL language. The VHDL-ICE, when analyzing a given DU, produces a library unit and automatically provides information regarding its transitive closure, i.e., the set of DUs on which it is dependent, and its anticlosure, i.e., the set of DUs depending on this particular DU. The closure and anticlosure information of a DU is also automatically maintained by the VHDL-ICE build management capabilities, see figure 7. The closure and anticlosure of DE is similarly defined regarding the corresponding DEs.

According to the LRM, for a given library logical name, the actual name of the corresponding DLs in the host environment may or may not be the same. Therefore the closure and anticlosure information of a given DU results to be an implementation-dependent one. To adequately solve this usually unknown VHDL feature for most of the existing VHDL tools in the market, the VHDL-ICE build management capability provides a friendly mechanism to associate a library logical name with the VHDL-ICE host-dependent library.

The VHDL-ICE, when viewing the information regarding the DLs content from the DEs perspective, also offers the possibility of automatically produce a Configuration Declaration DU corresponding to the design hierarchy static elaboration of each DE. This DU, together with the closure and anticlosure information of all DUs and DEs, can be considered as the equivalent to the bills of materials that document software system builds, and can completely and reliably recreate the environment of any build.

The VHDL-ICE Assessment facilities include the possibility of navigating through the block hierarchy of a DU, showing a lot of useful information created after the DU analysis process, through the design hierarchy defined by a DE and created in the DE elaboration process or through the simulatable VHDL processes network, obtained from the initialization of the elaborated model. The debugging facility allows to navigate upon the dynamic simulation model, using three different views : source code, components hierarchy and simulatable code. Finally, the VHDL Style give the possibility of verifying if a given set of DUs or DEs satisfy different quality rules, proposed as a guideline to the design process. The collection of rules applied is flexible (selecting or deselecting rules from the whole available set) and it is also possible to vary the priority level they are checked with.

4. Conclusions and Future Work.

The availability of the Teamwork ,Organization Units, Workspace and Build Management tools, integrated in the VHDL-ICE environment, provides benefits such as reducing engineering design time, reduced time-to-market, and increased VHDL product quality and IP commercialization. For most organizations in highly competitive industries such as the ones using VHDL based methods and tools, these benefits are too great for this technology to be overlooked.

VHDL-ICE can be considered as a pioneer of a new kind of VHDL tools supporting a system level methodology based on IP commercialization. Current version of VHDL-ICE is implementing the first of the three expected capabilities described in subsection 3.1. This VHDL-ICE version already presents the advantages offered by a concurrent and distributed implementation of a common environment in comparison with state of the art VHDL environments currently on the market. However, the other two expected capabilities will definitely increase the gap between the existing approaches and the one presented in this work.

Next step will increase the VHDL models management and their IP commercialization. Future work will also be focused on AIRE/CE and VHPI standardization activities for increasing VHDL-ICE tools integration and providing an open solution for Workflow and Design Process Control.

References.

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

[2] S. Olcoz, Lorenzo Ayuda, Ana Castellví, María García, Iván Izaguirre, Olga Peñalba, “Implementing a VHDL Design Manager: VHDL-ICE.” VHDL International Users’ Forum, Spring Conference, April 1997, pp. 93-102.

[3] S. Olcoz and P. Menchini, “HDL Interoperability: A Compiler Technology Perspective”, VHDL International Users’ Forum, Fall Conference, October 1996, pp. 51-58.

[4] H. Sahm, C. Mayer, J. Pleickhardt, J. Schuck, S. Späth, “VHDL Development System and Coding Standard.” Proceedings of the 33rd Design Automation Conference, Las Vegas, NV, 1996, pp. 777-782.

[5] B. Meyer, “Eiffel: The Language”, Interactive Software Engineering, Inc. July 1989, also published by Prentice Hall.

[6] S. Mullender, “Distributed Systems”, Addison-Wesley Publishers, 1993.

[7] 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, October 1996, pp. 33-42.

[8] Francoise Martinolle, Debra Corlette, Sathyam Pattanam, “A VHDL Procedural Interface for VHDL: VHPI”, IEEE/VIUF International Workshop on Behavioral Modeling and Simulation, Washington, October 1997.

[9] Stuart I. Feldman. “Make – A Program for Maintaining Computer Programs”. Software – Practice and Experience, 9 (4), pp. 255-265, April 1979.

[10] Clovis L. Tondo, Andrew Nathanson and Eden Yount. “Mastering Make: A Guide to Building Programs on DOS and UNIX® Systems”. Prentice-Hall, Inc., 1992.