Proceedings of the 1999 Winter Simulation Conference
P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, eds.

WEB-BASED ANALYSIS AND DISTRIBUTED IP

Philip A. Wilsey

Clifton Labs, Inc.
Cincinnati, OH 45241, U.S.A.

ABSTRACT

The web presents an opportunity for realizing a distributed design framework supporting multi-disciplinary, multi-organizational collaborative design and analysis activities. The potential for deploying online, reusable parts libraries for virtual prototyping and design analysis exists. However, several issues must be solved before vendors will be willing to provide online access to their intellectual property (IP). This paper reviews the main problems facing the web-based design and analysis community before the successful application of web-based virtual prototyping can become a reality. To amplify and solidify our arguments, the application domain of web-based hardware/software co-design is used.

1 INTRODUCTION

The interim report from the President's Information Technology Advisory Committee (Interim Report 1999) has identified component-based software development and a scalable information infrastructure as the key research issues for developing next generation software. It is clear that a multidisciplinary approach in defining new application programming interfaces is required for component-based development of global-scale software systems. In addition, models need to be developed for predicting the performance of dynamic and adaptive systems. However, the chief problem prohibiting fast and effective virtual design, prototyping and component reuse of global-scale software systems is the absence of available, distributed design environments for design, including the supporting infrastructure for web-based collaborative design activities (Page, Griffin, and Rother 1998). However, to ensure the successful, widespread acceptance of web-based design activities, serious steps to ensure the protection of Intellectual Property (IP) is essential.

Concerns for preserving IP make licensing and accessing detailed design data for virtual prototyping difficult. In general, vendors are unlikely to provide detailed specifications of product data unless their (legitimate) concerns regarding the protection of the product data from abuse and unintended distributions are adequately satisfied. Addressing these concerns can lead to complex, protracted negotiations that require detailed, project specific, licensing agreements. For example, within the electronics industry, numerous companies are developing and selling IP (electronic subassembiles, *e.g.,* JPEG circuits, memory chips/cells), but are doing so in a completely off-line manner. A more ideal solution would have the IP provider distribute their product data on the web for testing with virtual prototypes and eventual integration into marketable systems.

This paper describes our work to develop the web-based design technology necessary for vendors to safely distribute and protect their IP on the web. The basic strategy is to provide non-proprietary interface definitions on the web and to retain detailed product performance data behind protected firewalls. The public interface definitions enable interconnection with other regions of a virtual prototype and distributed analysis capabilities allow an evaluation of the fully operational virtual prototype. Thus distributed simulation across the web becomes feasible, and concerns about the security of proprietary data are addressed. IP is protected behind a firewall and distributed simulation is performed with the IP data wholly contained behind the firewall — only simulation interface data denoting changes to interface data values is exchanged over a potentially public network.

2 THE DISCOE PROJECT

To address concerns for preserving IP, Clifton Labs, Inc. has embarked up the development of a distributed design environment for hardware/software co-design. The DISCOE (DIstributed Simulation COllaborative Environment) project encapsulates a number of critical technologies required to provide a web-based distributed simulation environment emphasizing the preservation of Intellectual Property.

As previously stated, the chief problem addressed by the DISCOE program is the creation of a distributed design environment for hardware/software co-design, including the

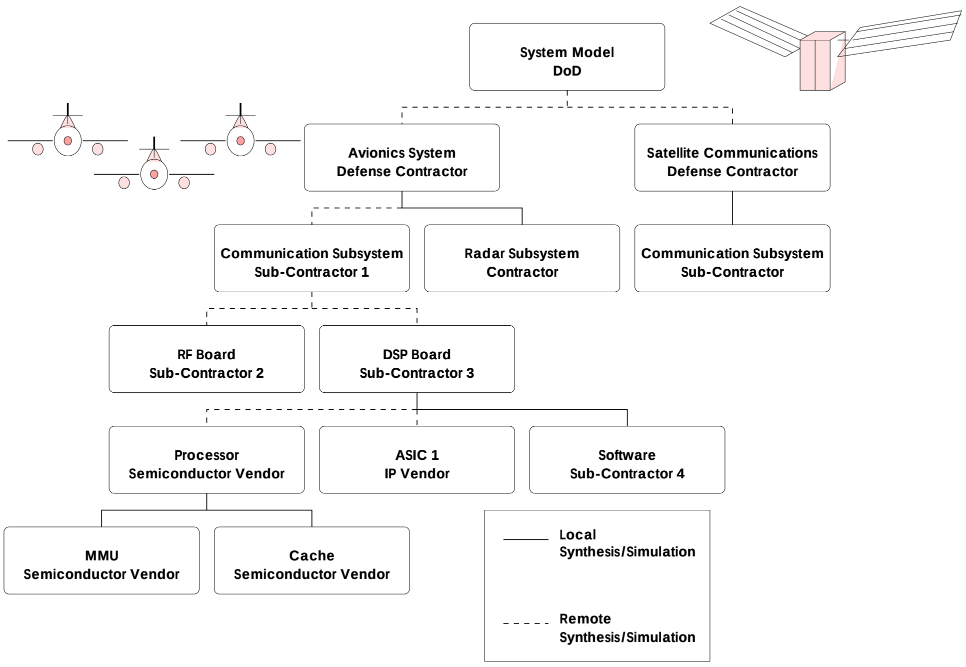

Figure 1: Distributed Architecture of a Communications System

supporting infrastructure for web-based modeling and simulation activities. This infrastructure enables a level of collaboration between designers of electronic hardware and software systems that was previously difficult to achieve, if not impossible. Today, multiple design teams within the same organization may develop components for a single system, and other components may be IP provided by a third party. In either case, the parties participating in the design must collaborate in order to fulfill the design goals. Furthermore, these parties may be geographically dispersed, and may have sensitive design information to protect.

For example, consider the development of a large-scale air/space communications system (Figure 1). Figure 1 shows the architecture of a communication system, that would support an avionics operation, where the components that make up the system are distributed across a number of organizations. Each organization may be reluctant to release their Intellectual Property for integration and simulation. The DISCOE concept allows these components to remain at their respective sites, yet provides support for a complete virtual prototype through local and remote simulations. As shown in Figure 1, the digital signal processing (DSP) subsystem is composed of a DSP core, a custom ASIC, and supporting software. It may not be feasible for the DSP Board Subcontractor to release a simulatable model of the digital signal processing subsystem if the model contains sensitive design information regarding the ASIC. Typically, the DSP Subcontractor would be forced to develop a simulatable model that would not reveal the Intellectual Property of the ASIC vendor. The idea of developing another simulation model for a system which is already simulatable is not easily accepted, but is usually entertained because it is the only means of providing a solution using existing technologies, while protecting individual IP information.

Figure 1 introduces another problem encountered during the integration of a system or a system of systems. Integration is difficult because the components may be distributed both geographically and throughout multiple organizations, and over multiple tools. In this situation, organizing and maintaining up-to-date repositories of these components can be extremely difficult, especially during the development process. Disperse design information is usually the result of collaborative design activities. Each of the subcontractors depicted in Figure 1 is collaborating within the design of the overall avionics system, although each participant may not be acutely aware of his part in the overall design.

Figure 2 shows an example of a collaborative design activity as supported by the DISCOE project. Two design teams, Design Team A and Design Team B, are collaborating to create a system. Design A utilizes a component maintained by IP Vendor X, while Design Team B has subcontracted a portion of its design to Subcontractor Y. The figure shows how the two designs depend on external components supplied by two distinct organizations. Design Team A is evaluating the use of Component x, supplied by IP Vendor X, within its design. The online availability of Component x, and the ability to simulate Component x within a virtual prototype of Design A, allows the de-

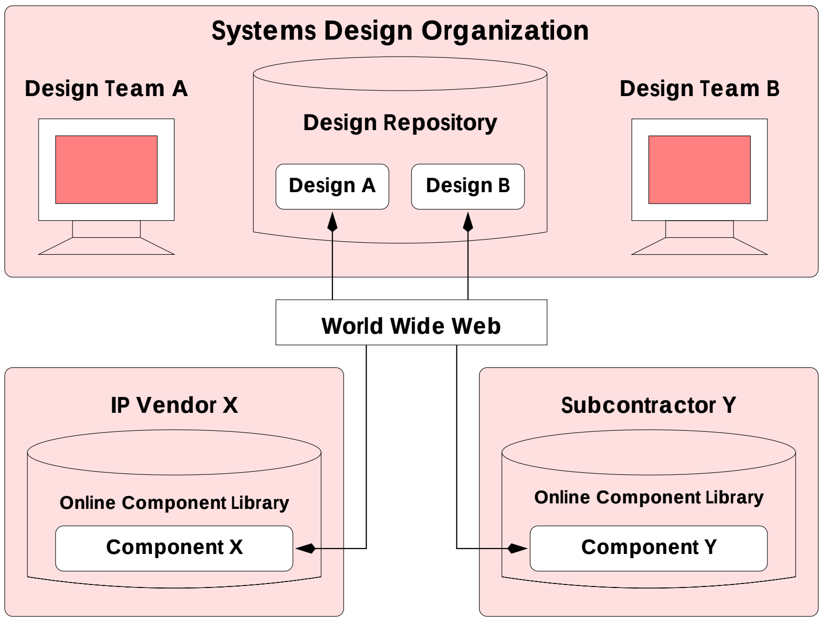

Figure 2: Distributed Collaborative Design Activity

sign team to perform Simulation Based Acquisition (SBA) without entering into a formal agreement with the IP Vendor.

Figure 3 depicts the simulation of the overall system design. Notice that only simulation information is passed between the Systems Design Organization, the IP Vendor, and the Subcontractor. Each participant in the distributed simulation is aware of only the external interfaces of the other participants. This ensures that Intellectual Property contained within the inner workings of the design remains invisible to the outside world. Another problem encountered within distributed simulations is the management of interactions between heterogeneous simulation models. Some models may be written in “C”, VHDL, Spice, or some other executable form. This can lead to problems when attempting to connect these models to form a virtual prototype of the system as a whole. DISCOE’s solution to this problem is a heterogeneous simulation backplane that supports a wide variety of simulation models.

This section has introduced many of the problems faced while developing today’s large-scale systems, and has proposed distributed simulation environments as a possible solution. A distributed simulation environment can be utilized to solve another crucial problem, that of Intellectual Property protection. In Section 3, we will discuss the architecture of the DISCOE system, and its relevance to Intellectual Property protection. In Section 4, we expand on the ideas of IP protection and discuss the preliminary results of the DISCOE program. Finally in Section 5, we conclude the discussion of IP protection through distributed simulation and discuss future efforts of the project.

3 THE DISCOE ARCHITECTURE

The chief problem prohibiting fast and effective virtual prototyping and component reuse is the absence of available, distributed design environments for complex system design, including the supporting infrastructure for web-based modeling and simulation activities. Multi-disciplinary, multi-organizational collaborative design activities are poorly supported by current design tools. Furthermore, the need for online, distributed, competitively licensed IP component libraries has received little recognition. A collaborative design environment for building today’s complex systems and components requires the following capabilities:

- An ability to support the deployment of searchable, distributed, online component libraries that preserve Intellectual Property (IP) (Fishman 1998; Lee 1997) rights and promote multi-vendor support of common parts for competitive costing. The structural and behavioral properties of online parts must be available for virtual prototyping and yet must preserve IP.
- Distributed compilation, simulation, and analysis (including synthesis (De Micheli 1994)) of systems designed with multi-vendor design data. To protect vendor IP, the simulation and analysis of remote data must have the ability to occur remotely on vendor supplied computational platforms or through IP protective APIs such as OMI (Open Model Forum), VSI

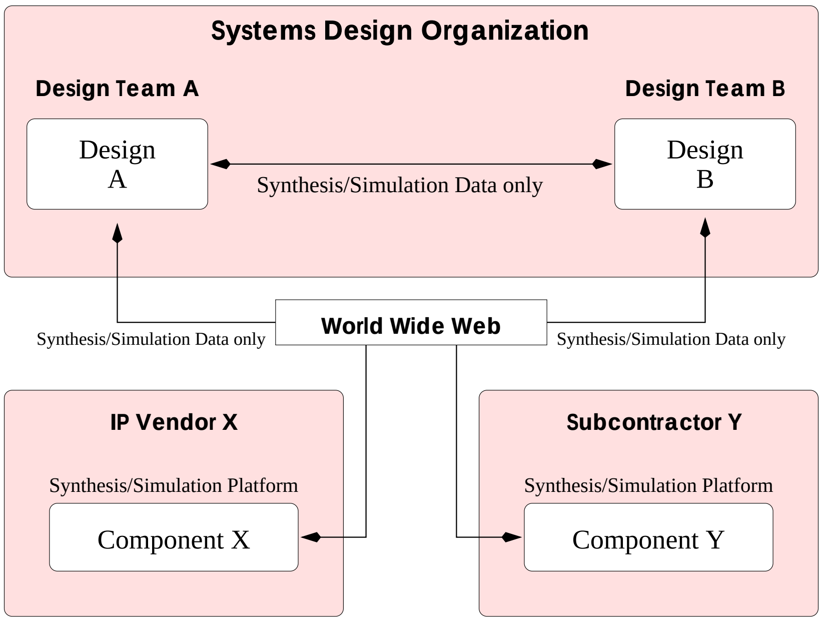
Systems Design Organization
Design Team A
Design Team B
Design A
Design B
Synthesis/Simulation Data only
World Wide Web
Synthesis/Simulation Data only
Synthesis/Simulation Data only
IP Vendor X
Subcontractor Y
Synthesis/Simulation Platform
Synthesis/Simulation Platform
Component X
Component Y

Figure 3: Distributed Synthesis and Simulation of a Design

(Alliance), or AIRE (Willis, Wilsey, Peterson, Hines, Zamfriescu, Martin, and Newshutz 1996).

- A portable design language for expressing the structural and behavioral properties of components to be used in hardware/software systems and components. There are several candidate languages for this activity, namely VHDL (IEEE 1993), SUAVE (Ashenden, Wilsey, and Martin 1998), VSPEC (Baraona, Penix, and Alexander 1995), and Java (Gosling, Joy, and Steele 1996). Recent research on languages for modeling multiple domains at multiple levels of abstraction has further resulted in the development of new multi-level and multi-model languages such as VHDL-AMS (IEEE Computer Society 1997) and object-oriented physical modeling (OOPM) languages such as DML/CML (Fishwick 1998a; Fishwick 1998b).
- A database maintaining design data that supports concurrent sub-team design activities across distinct sub-regions of a design. In particular, the database must support collaborative design activities involving two or more teams working on a single design. The database must also support design integration, versioning, and change logs.
- The hardware/software co-design environment itself must also be online and accessible remotely (with controlled access to protect the IP of the CAD tool vendor). Support for heterogeneous execution and platform independence is also necessary.

The DISCOE project is developing a design environment to support distributed, collaborative design activities. The environment supports secure transmission of design data by remote analysis and simulation of vendor protected IP. The analysis and simulation are platform independent; they operate in a distributed, heterogeneous web-based computational platform. A searchable online library promotes design reuse and relieves the effect of parts obsolescence. Furthermore, the organization of DISCOE supports future expansion for other analysis capabilities such as co-synthesis, and support of HLA (Dahmann, Fujimoto, and Weatherly 1997) and man-in-the-loop simulation.

3.1 The Major Components of DISCOE

Figure 4 depicts an instantiation of the DISCOE architecture in which a single Systems Designer and a single Intellectual Property vendor are collaborating to produce a virtual prototype of a design. The DISCOE architecture utilizes a set of critical components that support distributed collaborative design activities while ensuring the Intellectual Property rights of each participant. DISCOE provides this support in the following way:

- A searchable online Intellectual Property component catalog
- A heterogeneous simulation backplane that supports distributed synchronization.

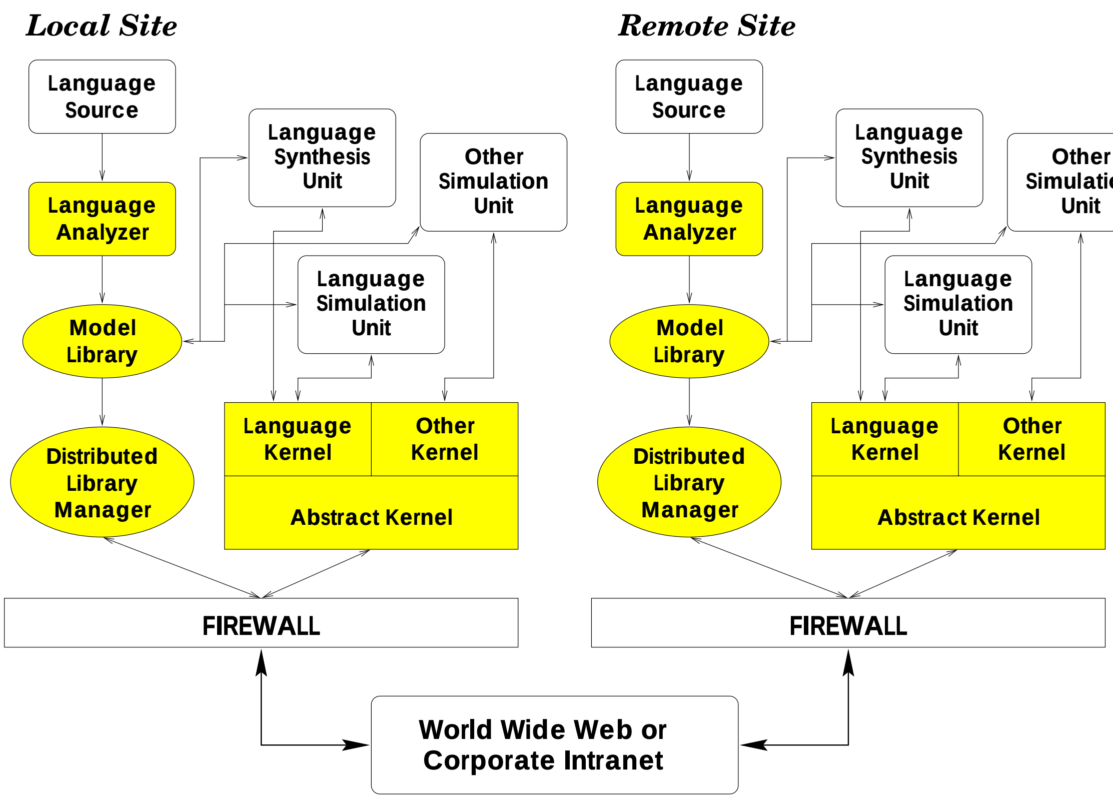

Figure 4: The DISCOE Architecture

- A secure Intellectual Property distribution model
- A Java-based Hardware Description Language (HDL) front-end that analyzes a component model into the IIR memory-resident data structure of the AIRE/CE standard. This analyzer is used to create user and vendor component libraries (OMI, VSI, or FIR compliant).
- A suite of Java tools that is dynamically distributed to support the distributed design and analysis activities.
- Online IP Component Catalog

Online access to IP components is critical to the collaborative design process. These IP catalogs provide the engineer with a wide choice of IP components that provide potential solutions to the design problem. Over the past twelve months, a number of IP catalog sites have appeared on the world wide web. These web sites do an excellent job of presenting the engineer with a collection of potentially useful components. Some provide detailed technical documentation, and even simulatable models. However, none of these sites has achieved the "try before you buy" concept so widely prevalent on the Web. Although some sites do provide access to simulatable models, these models must be downloaded to the local site in order to be integrated within the prototype. Many of these models are only high level behavioral models, while others are fine-grained simulations protected by what the industry calls "encryption through compilation". In the first case, the consumer of the IP is put at a disadvantage, because he may not be able to simulate the component with the desired accuracy. In the second case, the IP vendor is put at a disadvantage, because by allowing his model to be downloaded he has essentially lost control of his intellectual property.

DISCOE addresses these problems by merging the concept of an online component catalog with a distributed simulation environment. Intellectual Property possessed by the IP vendor remains securely behind the firewall. The only access to this information is through the distributed simulation backplane, and the presentation of the IP Interface Specification on the web browser. The systems designer browses online IP component specifications, including technical data, application notes, and other appropriate information. Thus far, things are no different than an online Data Book. However, when the designer finds an appropriate component, he may then download the Remote Model Interface for the said component. This interface does not contain any design information other than the external stimuli available on the component. The Remote Model also possesses the ability to connect to the real IP Simulation Model, which remains securely nestled behind the firewall, by using the distributed simulation backplane.

3.2 Heterogeneous Distributed Simulation Backplane

Note that the DISCOE simulation environment is referred to as both distributed and heterogeneous. This distinction is made, because there are two very important properties required for a successful web-based simulation environment. First, the simulation backplane is distributed, because the simulation models participating in the virtual prototype are likely to reside in not only different geographical locations, but different 'political' organizations. The ability to leave Intellectual Property in its original location, yet still retain access to it is most beneficial to both parties involved in the transaction. Second, the simulation backplane is designed to be heterogeneous. A web-based design environment that required all participants to model and simulate using the same set of tools and modeling paradigms, would be just about as useful as requiring all the drivers in the world to purchase motorcycles, because we could save money on asphalt and double the number of lanes on the highway. The heterogeneity of the backplane allows IP vendors to provide models in their most appropriate form. The only catch to this apparent benefit, is that the Remote IP Interface must be able to plug into the designer's local simulation environment. Until the formalization of simulator interfaces is completed, which appears to be in the near future with the advent of HLA for DoD simulations, as well as the Open Model Interface (OMI) for the Electronic Design Automation industry, DISCOE is relying on a conversion of simulation interfaces into a standard format known as the heterogeneous simulation backplane BACKPACK.

Figure 5 depicts the BACKPACK distributed simulation backplane. Note that all of the models which wish to participate in a simulation are eventually interfaced with a Common Object Request Broker Architecture (CORBA) layer. CORBA is a standard that allows distributed objects to interact in a network-based environment. Thus, at its lowest level, the BACKPACK backplane consists of a collection of distributed objects representing ports, signals, and controls.

3.3 Analyzer and IP Publishing

In Figure 4, one will note the appearance of an Analyzer that feeds into the design repository on the system designer's portion of the diagram. This analyzer, being developed as part of the DISCOE project, extends the abilities of the VHDL language to support concepts required for high-level modeling and simulation. The analyzer accepts the input language SUAVE, which is a superset of VHDL and provides extensions described in the following paragraphs:

The introduction of VHDL into the design flow for digital electronic systems has enabled faster design turn-around and a reduction in the number of design defects. By capturing the intended behavior of a system, VHDL enables simulation of a design before prototype fabrication, allowing defects to be detected earlier in the design flow. It also allows automatic synthesis of the design into a physical realization, eliminating and reducing the opportunity for introducing further design defects.

The current state of the art involves expression of a design using VHDL at the register-transfer level (RTL). With an RTL design, the designer can verify functionality through simulation. The design is then synthesized to gate-level net-list, which is simulated again to verify correct synthesis within required performance constraints. While this is a significant improvement over previous design flows, there is room for improvement.

Efforts such as the RASSP program have sought to improve the design flow by concentrating on design expression at the system level. What is missing at the register-transfer level is a means of expressing design intent at a high level of abstraction that can be analyzed, simulated and automatically synthesized to lower levels of abstraction (and physical realization, hardware or software - co-synthesis). It is desirable that the means of expressing system-level design intent be semantically integrated with lower-level hardware and software design representations to avoid the "semantic gap" that otherwise occurs.

SUAVE is one attempt to achieve the goal of extending VHDL to make it more readily express system-level design intent. In particular, SUAVE includes the following extensions:

Data Modeling: SUAVE proposes object-oriented and genericity extensions to VHDL. Since VHDL's type system is based on that of Ada, the approach is to adapt language mechanisms developed in Ada. The "programming by extension" approach involves defining extensible ADT's as tagged records and subprograms in packages. The type derivation mechanism allows new ADT's to be derived, inheriting and augmenting operations defined for the parent ADT. Use of extensible ADT's in this manner allows the system-level design to separate the concerns of data modeling and system architecture, thus making the system-level task more manageable. The genericity mechanism, also adopted from Ada, is generalized to apply to design entities, allowing parameterization not only of software components, but also of structural components. This affords the designer significant opportunity for reuse, not only at the system level, but at all levels of abstraction.

Abstract Concurrency and Communication: SUAVE extends VHDL by introducing a communication channel that allows asynchronous message

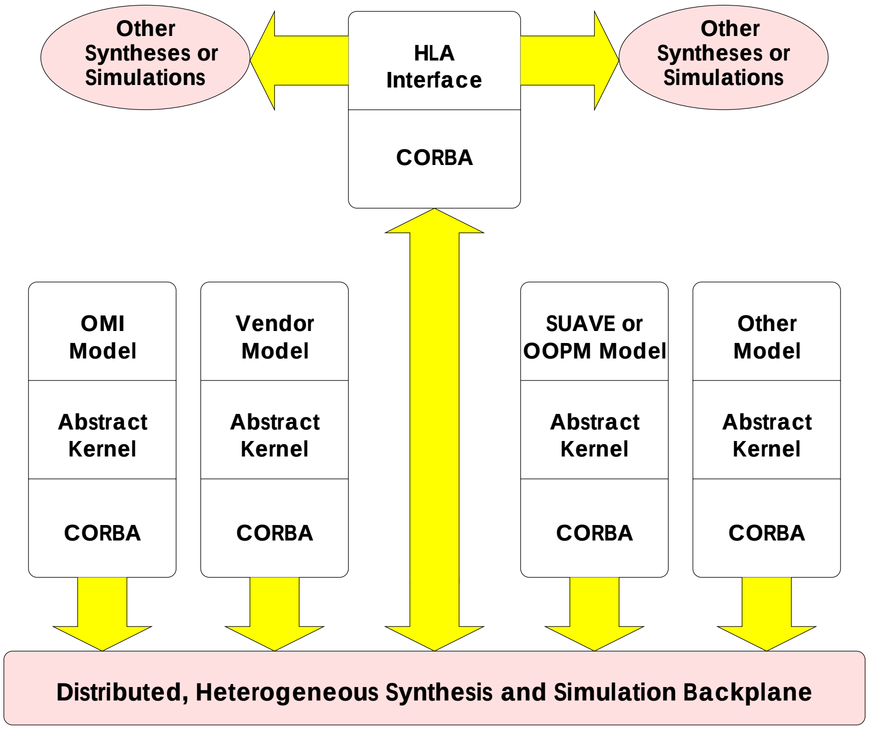

Figure 5: Distributed Simulation Backplane (BACKPACK)

passing between processes. Channels are an abstraction of signals, and allow expression of communication without details of electrical characteristics and strict timing. Channels can be statically created, or dynamically created to communicate with dynamically instantiated processes.

SUAVE generalizes the concurrency model of VHDL, by allowing declaration of process types that can be either statically or dynamically instantiated. A declared process includes an explicit interface, consisting of signals or channels. A declared process can also be generic, allowing type parameterized process that can be reused for dealing with different types.

Dynamic Structure and Reconfigurability: SUAVE extends VHDL to allow expression of reconfigurable structures by generalizing the block model of the language. Currently, a block is a static partition of an architecture body, encapsulating processes and component instances. SUAVE generalizes this by allowing the declaration of block types that can be statically or dynamically instantiated. Static instantiation is similar to static specification of blocks, but allows a block type to be reused in several places in a design. Dynamic instantiation allows modeling of resources that are added to the system during operation, and avoids the problem of supposedly inactive resources affecting system operation.

The SUAVE analyzer has been specifically designed to allow the generation of Intellectual Property protecting component libraries. The generation of these libraries is referred to as IP Publishing. Other forms of IP publishing can be performed using existing tools. The concept of the SUAVE language has been presented here to address the modeling and simulation needs of a project, from initial high level behavior to lower level design.

3.4 Collaborative Design Environment

DISCOE provides a collaborative design environment in order to manage the complexity of large scale distributed design activities. The graphical design environment of DISCOE supports collaborative design activities with the following features:

Version Control: DISCOE provides revision control and configuration management for design entities. This information is managed using a client/ server architecture that gives remote design teams controlled access to design entities. DISCOE provides file locking, automated merging, revision logs, and versioning through platform independent tools and a friendly user interface.

Information Sharing: DISCOE allows distributed design teams to collaborate by sharing information through not only the distributed design database, but also using a number of other features. DISCOE users can query the design database, automatically receive notifications regarding changes, report and monitor design issues, and collaborate using shared tools such as whiteboards, real-time meeting rooms, and persistent discussion groups.

User Authentication: DISCOE protects sensitive design information with user authentication, and secure communications through encryption.

Graphical Design Capture: DISCOE provides a visual representation of the information contained within a design that allows the design engineer to easily comprehend the concepts of a design entity.

Intellectual Property Reuse: DISCOE supports IP reuse by allowing design teams to integrate external IP into their designs. DISCOE users may search multiple remote IP catalogs and select components that satisfy their design needs. The IP associated with remote components remains in the component provider's control.

Intellectual Property Distribution: DISCOE supports IP distribution by allowing users to package their designs for export as self-contained IP components. These components can then be placed on a Component Server that allows other design teams to reuse the IP.

4 INTELLECTUAL PROPERTY PROTECTION

In a distributed collaborative design environment, the providers of reusable IP must be assured proper credit when their IP is utilized within a design. However, to be fair to the designer, the IP provider must offer reasonable assurance that the IP component will satisfy the needs of the designer. DISCOE satisfies both of these requirements through distributed simulation and IP preservation.

DISCOE allows designers to instantiate remote components in their designs to test the compatibility of the remote component. DISCOE also allows the IP provider to control access to the IP component. During distributed simulation, a remote component is only visible through its external interface that is defined by the IP provider. Simulation event information is transmitted between the designer's local simulation environment and the providers remote simulation of the component.

The provider's interests are further protected through additional security measures. IP components reside on a component server that resides behind the provider's firewall. The design content of IP components is further preserved by placing only the compiled, simulatable representation of the component on the DISCOE component server. Access to the IP components is restricted through user authentication allowing the IP provider to control access to IP catalogs. Finally, the simulation data passed between local and remote simulations is transported on a secure communications link.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we have described the requirements of a distributed, web-based design and analysis environment, and addressed the issues of intellectual property preservation in such an environment. We believe that the envisioned collaborative environment satisfies these requirements in a unique fashion that fully supports the integration and preservation of intellectual property. The environment supports fast and effective virtual prototyping and component reuse. It allows teams of designers to collaborate at higher levels of abstraction. In addition, the environment helps designers locate appropriate intellectual property components and integrate them into their designs. Finally, it protects the rights of the intellectual property provider through distributed synthesis and simulation, encryption, and user authentication.

REFERENCES

Alliance, V. Virtual Socket Interface (VSI). `http://www.vsi.org`.

Ashenden, P. J., P. A. Wilsey, and D. E. Martin (1998, April–June). SUAVE: Extending VHDL to improve modeling support. *IEEE Design and Test of Computers 15*(2), 34–44.

Baraona, P., J. Penix, and P. Alexander (1995). VSPEC: A declarative requirements specification language for VHDL. *Current Issues in Electronic Modeling 3*.

Dahmann, J. S., R. M. Fujimoto, and R. M. Weatherly (1997, December). The Department of Defense High Level Architecture. In *Proceedings of the 1997 Winter Simulation Conference*, pp. 142–149.

De Micheli, G. (1994). *Synthesis and optimization of digital circuits*. New York: McGraw-Hill.

Fishman, S. (1998). *Software development: A legal guide* (2 ed.). Nolo Press.

Fishwick, P. A. (1998a, December). An architectural design for digital objects. In J. S. C. D. J. Medeiros, E. F. Watson and M. S. Manivannan (Eds.), *Proceedings of the 1998 Winter Simulation Conference*, Volume 1, pp. 359–365.

Fishwick, P. A. (1998b, April). Issues with web-publishable digital objects. In *Proceedings of the SPIE Aerosense Conference*.

Gosling, J., B. Joy, and G. Steele (1996). *The Java Language Specification*. Reading, MA: Addison-Wesley.

IEEE (1993). *IEEE Standard VHDL Language Reference Manual*. New York, NY: IEEE.

IEEE Computer Society (1997). *IEEE Draft Standard VHDL-AMS Language Reference Manual*. IEEE Computer Society.

Interim Report (1999). President's Information Technology Advisory Committee. `http://www.ccic.gov/ac/interim`.

Lee, L. C. (1997). *Intellectual property for the Internet*. Wiley Law Publications.

Open Model Forum. Open Model Interface (OMI). `http://www.eda.org/omf`.

Page, E. H., S. P. Griffin, and L. S. Rother (1998, April). Providing conceptual framework support for distributed web-based simulation within the high level architecture. In *Proceedings of SPIE: Enabling Technologies for Simulation Science II*.

Willis, J. C., P. A. Wilsey, G. D. Peterson, J. Hines, A. Zamfriescu, D. E. Martin, and R. N. Newshutz (1996, October). Advanced intermediate representation with extensibility (AIRE). In *VHDL Users' Group Fall 1996 Conference*, pp. 33–40.

AUTHOR BIOGRAPHIES

PHILIP A. WILSEY is President of Clifton Labs and is an assistant professor of computer engineering at the University of Cincinnati. He received a B.S. degree in mathematics from Illinois State University; and he received M.S. and Ph.D. degrees in computer science from the University of Southwestern Louisiana. He is a member of IEEE, IEEE Computer Society, and ACM.