

# Standards Based Architecture for Systems Verification

Victor Berman

Cadence Designs Systems  
270 Billerica Road  
Chelmsford MA 01824  
berman@cadence.com

## 1.0 Abstract

*This paper describes a standards based approach to providing the infrastructure needed to support systems verification. The issues of the model availability, protection of intellectual property, and the semantics of mixed language verification systems are addressed.*

## 2.0 Introduction

In the early 1980's when specifications for VHDL were being developed, the visionaries behind this effort understood the need for a broadly encompassing notation for accurately describing system behaviors. They also had the wisdom to understand that in the world of rapid technology change, it would be impossible to capture the specific characteristics of that technology or in fact to capture the rapidly changing design and verification methodologies.

Realizing these inherent limitations they required that the specification be made flexible and tailorable so that well defined extensions could be made to address technology and methodology changes within the basic framework of the notation. This approach has proved successful in certain areas such as the Multi-Valued Logic Standard 1164 and the draft VITAL standard 1076.4 as well as several others under development.

Other disciplines required for system verification and design have been more difficult to address within the basic VHDL framework. Examples of these disciplines are:

- Intellectual property protection in component models
- Interoperability with non-VHDL models
- Semantics of VHDL models in non-event simulation environments

In order for VHDL to remain relevant for current and future design and verification, it is imperative that solutions to these problems be found within the context of well defined, open standards which provide efficient and effective tools for the system designer.

This paper describes an approach to resolving these issues by combining the work being done by the Open Model Forum (OMF) with research on operational

semantics for language interoperability to produce an open, standards based architecture suitable for systems verification. It is shown how an implementation of the Open Model Interface (OMI) based on the draft standard PLI IEEE-1364 can be used to verify models written in VHDL, Verilog, and C in existing VHDL and Verilog simulators that support this PLI. It is also shown how this work can be extended to address the issue of common semantics for simulation and logic synthesis leading to an approach for defining interoperability for synthesis tools.

### 3.0 Model Availability

One of the keys to system verification is the timely availability of appropriate component models. The current channels for model dissemination suffer from a number of problems which tend to limit their effectiveness.

Developers of electronics products have a need to get timely access to information about the logic and timing of components in order to meet tight product market windows. Semiconductor vendors wish to have their silicon incorporated into these electronic products but also wish to protect their investment in the silicon designs. Thus, there is a need to deliver critical data (especially logic and timing) about semiconductor devices to electronics manufacturers. Given that there are a number of simulators which are used by different electronics manufacturers, the semiconductor vendors desire to reduce their support costs for model delivery by getting the EDA vendors to agree on a standard interface for model delivery.

Today, simulation models are delivered through one of the following methods:

- HDL source models
- Binary models which are simulator specific (e.g. RapidSim or compiled Leapfrog models)
- Binary models from third parties (e.g. LMG) which are simulator independent

Each of these methods does not fully address the needs of the marketplace. HDL source models will work on a class of simulators from various EDA vendors (i.e. either VHDL or Verilog simulators) but generally do not offer semiconductor vendors enough protection of their intellectual property unless they are encrypted (which brings with it numerous problems for export). So, when semiconductor vendors use this method, the model is usually lacking in some details concerning function or timing (e.g. bus-functional models) and do not provide the fully capable model which would best serve electronics manufacturers needs. Simulator-specific binary models involve a high support cost for the semiconductor vendor when multiple simulators must be supported and limit potential customers for the model to supported simulators. Binary models from third parties are also not ideal because their development requires transferring the needed information to the third party; any HDL models developed to implement the semiconductor device are not directly used. Also, dependence on third parties introduces

additional business relationship issues which may discourage free access and EDA vendor support because of competitive conflicts.

### 3.1 OMF Developments

In response to these issues, the Open Model Forum (OMF) was formed to drive the development of a standard interface to simulators for the delivery of component models. This development is based on two existing industry technologies:

- The Verilog IEEE Standard 1364 PLI VPI and;
- The SWIFT Interface Specification.

These two technologies are being used to develop a simulator interface and a model interface to meet the requirements specified by OMF [9].

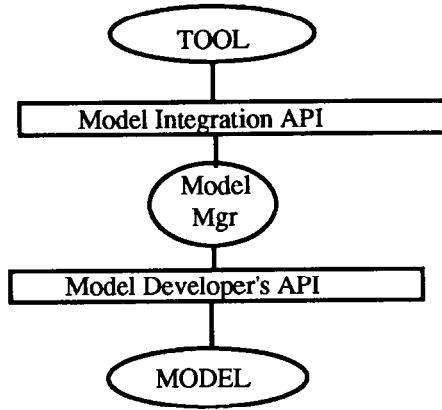
The Simulator Application Interface based on the SWIFT technology provides a means for existing simulators to provide access to models which adhere to this specification. This interface provides the traditional third party type of integration but will have the advantage of being an open standard once the OMI specification is published.

The Model Application Interface based on IEEE 1364 VPI provides a simulator/language independent API which can be used to guide the generation of models from existing VHDL or Verilog models using a model compiler. It is assumed that such model compilers will become available as the specification matures. Together, these two interfaces comprise the Open Model Interface (OMI).

### 3.2 Model Management -OMI

The purpose of this interface is to support the delivery of models in tool independent object code format. The communication between a tool and a model is, therefore, limited to set of procedure calls. The two components of the OMI may be thought of in terms of a Model Integrators API and a Model Developer's API. Each of these is seen (at least conceptually) as a piece of software supporting the interface. While the functions of these APIs is similar, there are differences in terms of the level of kernel services provided. This relation can be pictured as:

## Role of the APIs in the OMI Structure



**FIGURE 1.** The Role of the Model Manager/API

The model manager is responsible for handling any differences in semantics between the underlying simulation tool and the model. This may include kernel services which are not directly provided by the tool. The completeness of this layer will determine the limitations of the models which can be supported. The inclusion of these services directly by the simulator (or other tool) may have an impact on the efficiency of processing

### 3.3 Model Development

One of the objectives of this standard is to reduce to cost and difficulty of producing models. Figure 2, shows the view of model development based on existing models in the three supported OMI language, VHDL, Verilog, and C. The model compiler will take these languages as input and produce OMF compliant object modules suitable for shipping and use in an OMF compliant simulator for systems verification. Main modules such as the "Model" may be produced, or shared objects suitable for storage as a library element called by multiple modules may be produced.

### 3.4 Model Use

Models, developed independently by vendors from different categories (IC vendors, third party developers, EDA

vendors...) will be coordinated in a single verification environment by a model manager. Figure 3, shows such a configuration managed through the Model API.

### 3.5 Intellectual Property Protection

The protection of intellectual property in the distribution of models encompasses several key issues; business related, technology related, and legal. While object code is known to be decompilable to some extent, case law has generally held that compiled objects represent reasonable steps for protection of proprietary information. While encryption mechanisms arguably give more secure protection technically, case law covering encryption is not so clear as for compiled object. Furthermore, federal restrictions on the export of encryption technology makes its use for general model distribution undesirable at this time

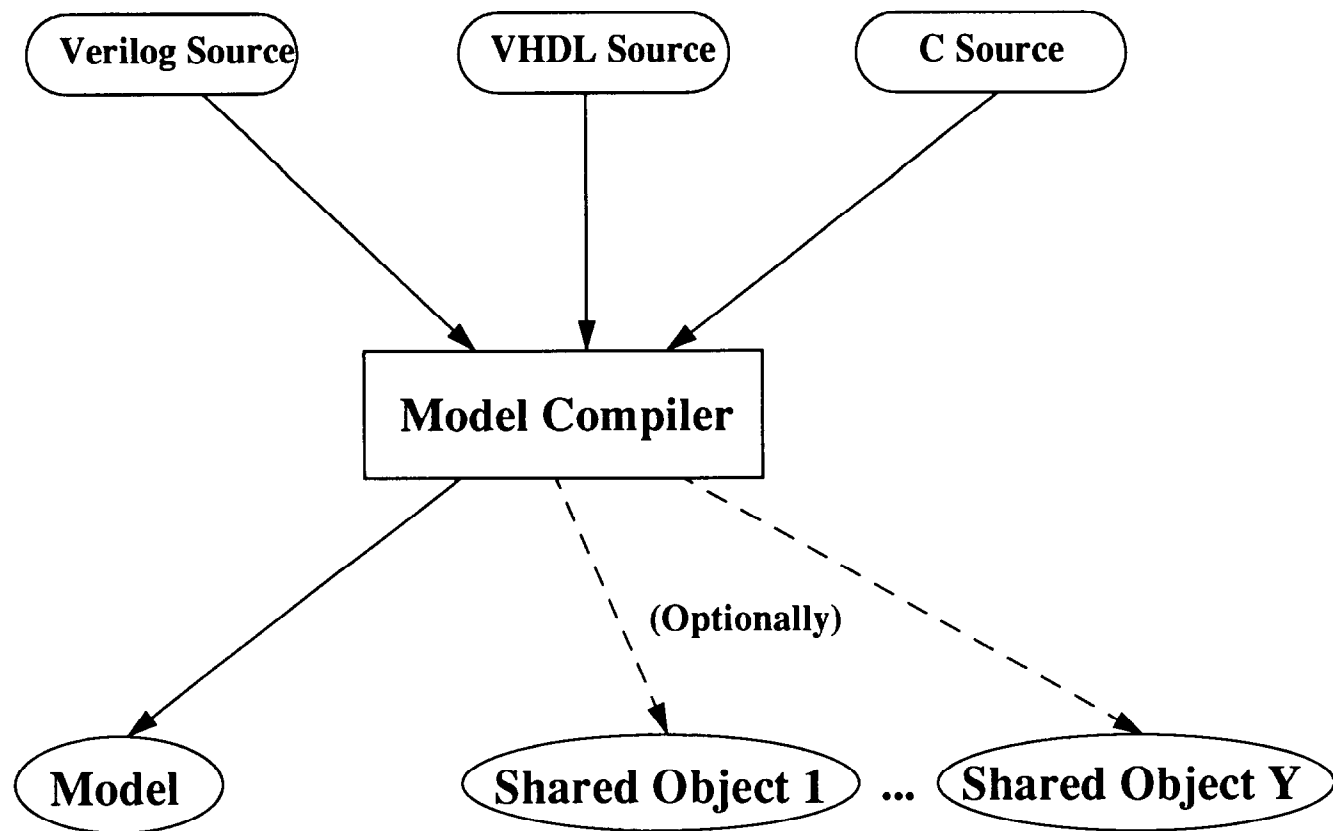
These considerations have led OMF in the direction of the use of compiled object as the vehicle for distribution. Since models will be used for different purposes in different phase of design, the specification must allow for tailoring the internal visibility into models. While a complete black box may be sufficient for regression tests at the system level, it is general necessary and desirable to has visibility into key registers and interface objects to monitor and validate system behavior.

To deal with this issue, the OMF specification develops the concept of a view port whose scope is explicitly defined by the model developer. This view port makes visible ports, parameters, registers and signals which are deemed necessary for viewing. All other objects are kept invisible so that information considered sensitive may be protected while the model is in use. While this ability to limit visibility combined with compiled object distribution provides a measure of protection, some vendors may not feel that it is sufficient and may choose to use legal means such as non-disclosure agreements to bolster security. Furthermore, tool issues such as options for generation of internal assembly code or other intermediate formats which may be reverse engineered must be guarded against.

Many of these issues would, from a technical standpoint, be better addressed by schemes for public key encryption. However, it will take a significant change in the US Department of Commerce rules for distribution of

## Fig 2. OMI Arch. (Model Development)

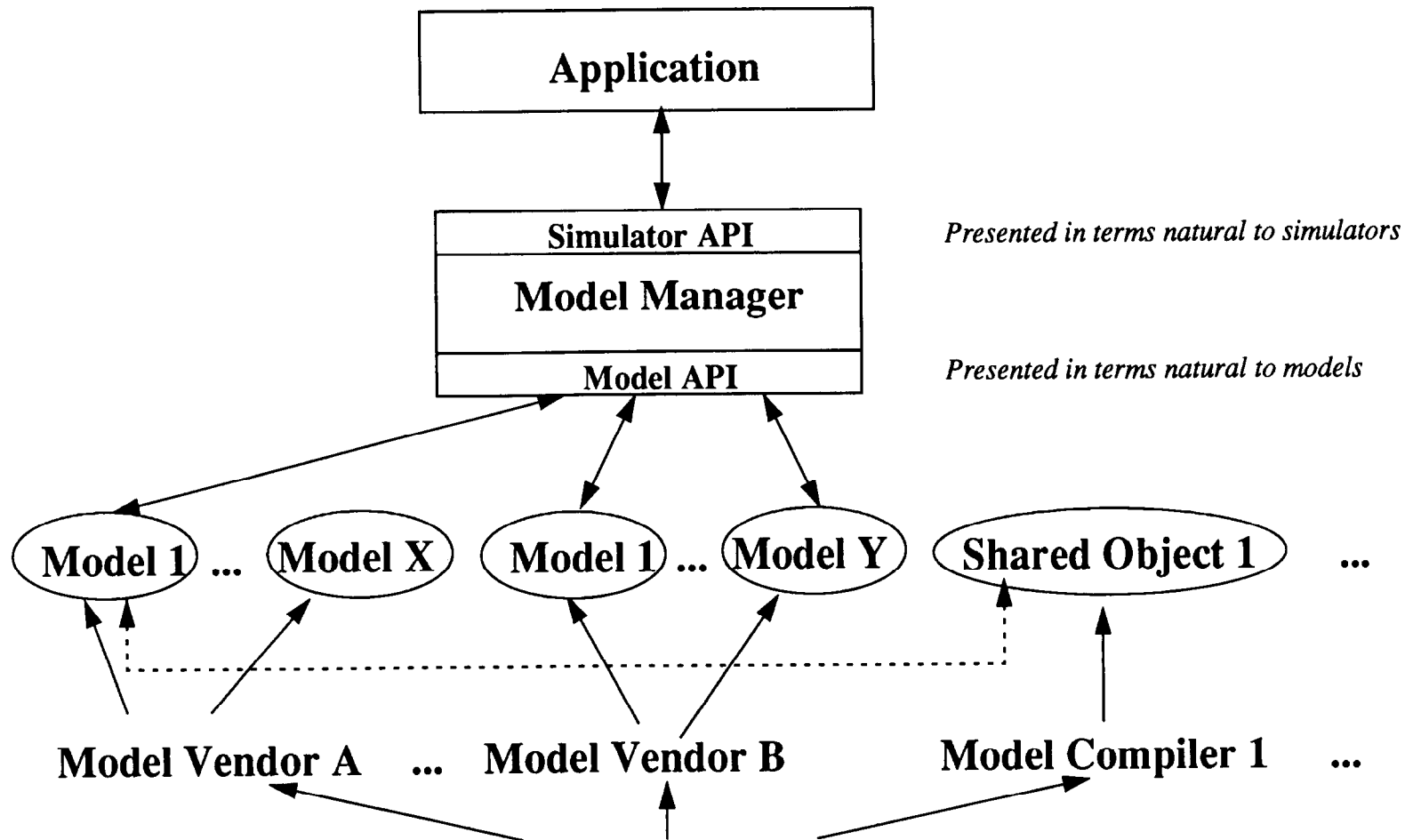
---



*Verilog, VHDL or C model source code may be used to create an object code representation of model suitable for simulation using the OMF protocol.*

---

Fig 3. OMI Architecture (Model Use)



encryption technology before this becomes a practical vehicle for commercial distribution of models containing significant intellectual property.

## 4.0 System Verification

### 4.1 Overview

The issues of model availability have traditionally been the major stumbling block to system verification. However, with the advent of multi-language system representations and the heavy reliance on logic synthesis tools in the design process and cycle simulators in the verification process. These factors have increased the importance of broadening the definition of HDLs (specifically VHDL and Verilog) from their event-driven, simulation centric paradigm, to one that is more meaningful in other domains and which deals with other language interaction.

### 4.2 Operational Semantics

Work in defining a more formal semantic for joint VHDL-Verilog systems has been underway for some time [1], [5]. The basis of this work is to form an operational definition of simulation semantics in terms of a series of event lists; how they are created, how events are added, how these event lists are updated, and how the evaluation of statements adds/removes elements to/from these lists.

This type of definition seems adequate for a range of event based HDLs but does not directly map into a solution for defining operational semantics for cycle based simulators or logic synthesis tools. In both of the latter cases, reasonable definitions seem to rely on satisfying a set of constraints at synchronous boundaries. This problem is related to so-called Formal Verification where mathematical derivations attempt to show equivalence between states. Work is needed to determine how such a system could be effectively used to provide a formal tie between verification through simulation and meeting constraints in logic synthesis.

While the commonality of semantics between logic synthesis and event simulation may seem tangential to the problem of system verification, closer analysis shows that without such a framework it is impossible to clearly define validation criteria across design boundaries that cross these domains. That is, if the results obtained from testing a model after a logic synthesis step do not match with those

expected based on event simulation there is no consistent decision process to determine which result is appropriate since there is no commonality of semantics between the two domains.

### 4.3 Synthesis Interoperability, Semantics and Verification

Logic synthesis is a specialized form of mathematical transformation which, loosely speaking, produces a model which is in some sense equivalent to its input model, but at a lower level of abstraction generally more suitable for physical implementation. Assuming that the input model is expressed in an Event-Driven defined HDL, the semantics of the input model are known only in terms of that simulation paradigm. All other information about that model such as the range of its allowed inputs, its timing characteristics under load or under varying physical conditions, are known, if at all, through notations external to the HDL. Therefore, the determination of the correctness of the synthesized model is at best known in terms of those external notations, and more probably only known anecdotally or through validation by simulation of a set of test vectors.

If one considers the typical verification process of a microprocessor based system the importance of this becomes clear. During the pre-silicon phase of the microprocessor development, high level models of the microprocessor are developed by the IC vendor to aid external system developers who are designing their application in parallel with the microprocessor development in order to meet their time-to-market windows. These high level models become the de-facto functional specifications used by the system designer. When these models or their RTL counterparts are further developed using logic synthesis the possibility of introducing tool specific, and unintentional changes to the functionality becomes probable since design constraints are introduced outside the domain of the defining language (HDL) of the higher level model. This does not imply an error in the logic synthesis tool so much as a gap in semantic information flow across the design step boundaries. The result for the system designer is that previous verification results based on the higher level model are not reliable negating at least to some extent the effort expended in performing system verification in parallel with the component development.

While other common, traditional, factors such as simulation speed and capacity limit the ability to adequately perform system verification, they are addressed in a continuum of improvements based on advances in simulation and work station technology. The more fundamental limitation due to semantic gaps require more specific remedies.

## 5.0 Missing Pieces - Future Work

The implementation and adoption by the electronic design community of VHDL represents a major step forward in providing a solid semantic base for electronic design. It is the first industry accepted notation for a broad level of design capture that is reasonably well defined independent of implementation and rich enough in abstraction to live beyond a short time-span of technology innovation. It however, is currently lacking in a few important areas which have been introduced in this paper. The affect of these limitations is to force designers to go outside of VHDL to complete their designs. Without either extensions to VHDL or interfaces to other recognized standards the result is a reliance on proprietary methods which are not consistent between different tools and tool vendors. This same situation with respect to ASIC sign-off capability was addressed by the VITAL initiative which has now resulted in a draft IEEE standard for modeling ASICs in VHDL [10]. Two of these are:

- Standard Procedural Interface
- Standard Logic Synthesis Methodology

Work has begun on both of these issues in two venues. The issue of Procedural Interface is being addressed by the OMF Technical Team. This work is based on the existing Verilog IEEE 1364 Standard and is being expanded to include VHDL semantics. The goal is to keep the user's view of the procedural interface as close as possible to the existing definition. The immediate goal is to support other-language models, but the end result will provide a standard

means of general communication with VHDL objects both statically and dynamically. This capability which has long been available in Verilog will encourage the development not only of models but also other tools for analysis and design which can be made tool/vendor independent based on the standard interface.

The other issue of logic synthesis methodology is currently in the early stages of study by the VHDL International Technical Activities Committee (VI-TAC). This problem, while challenging, is seen as the most important new bottleneck limiting the utility of VHDL for design.

## 6.0 References

- [1]Berman, Victor, "Standard VHDL Verilog Interoperability", Proceedings of the International Verilog Conference, 1994, San Jose, CA.
- [2] "IEEE Standard VHDL Language Reference Manual IEEE Std 1076-1987", 1988, The Institute of Electrical and Electronics Engineers, Inc. New York, NY.
- [3] "IEEE Standard VHDL Language Reference Manual IEEE Std 1076-1994", 1995, The Institute of Electrical and Electronics Engineers, Inc. New York, NY.
- [4] "IEEE Draft Verilog Hardware Description Language Reference Manual", April 1995, The Institute of Electrical and Electronics Engineers, Inc. New York, NY.
- [5]V. Berman, J.Lawrence. "Resolution of Simulation Cycle Semantics In a Mixed Verilog-VHDL Simulation Environment", Proceeding of the VHDL International User's Forum, Spring 1995, San Diego, CA.
- [6]Will Hobbs, "Model Availability, Portability, Accuracy", Proceeding of the Workshop on Libraries, Component Modeling, and Quality Assurance, April 1995, Nantes, France.
- [7]"Open Model Interface, Draft Version 0.3", Open Model Forum Technical Team, June 1995.
- [8]"SWIFT Interface Specification, Rev 1.0', Logic Modeling Cor., June 1993.
- [9]"Open Model Interface, Requirements and Rationale, Version 0.2", Open Model Forum, Technical Team, April 1995.
- [10]"Draft IEEE Standard VITAL ASIC Modeling Specification", July 1995, The IEEE Inc. New York, NY.