Interchange of Electronic Design Through VHDL and EIS

Richard M. Wallace
Computer Scientist
Air Force
Wright-Aeronautical Laboratories
Electronic Technology Division
Wright-Patterson, AFB, Ohio

Abstract

The need for both robust and unambiguous electronic designs is a direct requirement of the astonishing growth in design and manufacturing capability during recent years [1,2]. In order to manage the plethora of designs, and have the design data both interchangeable and interoperable, the Very High Speed Integrated Circuits (VHSIC) program is developing two major standards for the electronic design community. The VHSIC Hardware Description Language (VHDL) is designed to be the lingua franca for transmission of design data between designers and their environments. The Engineering Information System (EIS) is designed to ease the integration of data between diverse design automation systems. This paper describes the rational for the necessity of these two standards and how they provide a synergistic expressive capability across the macrocosm of design environments.

The Rational

The VHSIC Program has propelled forward the design density of electronic systems to a point where current computer aided design tools, design representations, and the corresponding data management systems begin to limit the designers' ability to design throughout the continuum of system levels to physical levels. In order to provide mechanisms for designers in the next decade, the VHSIC program has several design automation efforts under-way in its Technical Insertion and System Level Design tool subprograms. The dual focus of these subprograms is use of VHDL as the notation for design/description and the EIS for integration of design data. Initiation of the the VHDL program was motivated by the diversity of design notations that failed to encompass the broad range of descriptive capability required for advanced system documentation, and by the need of the DoD to provide a standard descriptive notation for systems that have life-spans upward of fifteen years. The VHSIC Hardware Description Language provides an economical method to decrease the system design time and cost of government re-procured ICs. Design costs for ICs are now in the range of $2 to $5 million and development costs must be reduced to meet future needs. Maintaining and upgrading electronic systems in inventory demands specific, current, and rigorous descriptions. As English can be vague, and fraught with idiomatic contextual references, the automation of design and design verification demands a technology independent, rigorous notation as is VHDL.

The EIS is envisioned to provide efficiency for the design process and to ease the insertion of VHSIC technology into electronic defense systems. To this end the development of an integrated design, documentation, and life-cycle maintenance system for complex electronic systems must support initial specification design data capture to fabrication and testing data in one continuum. An EIS system is not, and would not, be available from the commercial sector due to the high cost of development for a "turn-key" system being beyond most businesses. Therefore design automation tool users are forced to integrate an assortment of design tools from other vendors and those that are developed internally. The unique, proprietary, and internal design representations of each vendors' design automation tool complicates the integration task drastically. Integration has been a sever problem [3,4] while integration is known to be beneficial [5,6]; thus the EIS has as its main goal the reduction of the present difficulties involved with intgration of different vendors' design tools by developing a set of inter-oper-

ability standards and then demonstrating them. The VHDL in total is to be used in the EIS system as the design documentation formalism required of complex electronic systems.

VHDL -- The Lingua Franca

A standard hardware description language benefits all industries that depend on electronics. By its use the problem of a second source can be greatly reduced. But how is this accomplished? What makes the VHDL such a beneficial notation for electronic design? From the inception of a standard hardware description language [7] the focus of the language was to allow a hierarchical continuum of design notation from the system to gate level. A discussion of the language hierarchy must begin at its basic building block, the design entity; and then progress through its other features to show the capability of the language for electronic design and the the transfer of that design from designer to designer.

The design entity is the principal hardware abstraction in VHDL. A design entity provides the separation of interface and function to allow a hierarchical design decomposition. The crux of the design entity is the interface which allows the entity to be combined with other components. The interface is the abstraction's "pin-out" that describes the data paths and other factors that need to be known by component users. The secondary part to a design entity is its body which describes the organization and/or operation of a component. As an abstraction, the entity interface may possess multiple bodies, each representing a different implementation or emphasizing a different view of design. A design entity models electronics of any intricacy. Examples would be a logic gate, a flip-flop, a control unit, or a computer system. In fact, the range is only limited by the imagination of the designer as design entities can be used to describe any physical object having a bounded identity.

The design entity interface contains information that is common to the bodies that use the entity interface. This information includes data that is visible and is not visible externally. Of the visible data there are two types ports and generics. The non-visible information may define types, constants, and attributes that are used by the alternate bodies of the entity. Inclusive of the object information, the interface can contain assertions that specify operating properties and operational circumstances of the entity. Operating properties specify desired timing or functional relationships demonstrated by the entity. Operating circumstances specify external conditions that must be stated in order for the entity to correctly model its component.

To define communication channels among design entities and the outside world, the port data describes the mode and type of that information. To pass data that is not part of an entity's port interface, but is important to the operating circumstances, the design entity interface may have generics. For example, a generic value would be passed to the entity to specify a particular technology that the design entity is representing. Generics may represent instantiations of preconditions for execution.

Given the interface of a design entity, the designer provides a body that will describe the function of the entity. In VHDL there are two major divisions of entity bodies; the architectural body that expresses the data transformations that occur within the entity and the configuration body that controls the choice of design entities that are used to model sub-components and the distribution of signal definitions.

In an architectural body the description styles that designers use roughly fall into three categories: structural, data-flow, and behavioral. As is implied, structural description is approximately equivalent to the schematic connection of electronic components. The data-flow description method consists of register transfer level data transforms. The behavioral method of description allows the designer to specify transforms in wholly algorithmic terms. Any given architectural body may use these three general forms of description interchangeably.

With the capability of developing a library of similar component designs, it is desirable to make use of existing entities even if names or ports are not exactly what are required, but a subset interface will suffice. Additionally, a design series can have multiple configurations, each using slightly different design entities to implement the given component's behavior. The configuration body, which contains the configuraton specification, provides the ability to respecify the default association rules so that an architectural body's compo-

nents may be bound to corresponding but not identical design entities. The architectural bodies must preceed the configuration bodies which use them. In this way a configuration body can add or modify the enity configuration post-design without altering its basic architecture.

With the basic structural elements of the VHDL identified, the data of such a block structured language must follow in rigor, and scope. The VHDL is a strongly typed language based on the syntax and semantics of Ada. With this being known, the VHDL supports descriptions of objects from the typical bit values of '0' and '1' to higher levels of abstraction such as "integer," "message packet," and "instruction." With the range of data that can be described, VHDL avoids the pit-fall of predefining data types available to the designer. This gives the designer the ability to completely describe new data types as they are needed. The set of types available to the designer include predefined types such as BIT, BOOLEAN, REAL, INTEGER, CHARACTER, and TIME. Additionally all scalar and composite types are allowed. These types would include enumeration types, physical types (allowing expression of measurement defined in a base unit), records, and multidimensional arrays. VHDL has the ability to create functions and procedures and place these in packages to enable the designer to encapsulate algorithmic behavior.

The two most salient features of VHDL for future application to artificial intelligence are the ability to create and attach attributes to objects and have liturgical assertions that have global scope for a design entity. As new technologies emerge for design and construction of electronic circuits, VHDL provides an attribute mechanism that allows designers to associate extra information with descriptions of components or parts of components. As attributes can be referenced in VHDL this allows entities and data-objects to have LISP-like atom properties. This capability is useful in intelligent silicon compilation [8,9,10]. In order to produce designs that are both efficient, and more importantly correct, the VHDL has the assertion ability required in many verification systems [11]. Assertion statements check static or dynamic conditions that are either checked prior to simulation or during simulation as signal values change. Assertions may occur at any point in a VHDL description and are user controllable in order to report the condition of the entity.

EIS -- The Pax Romana

In 1984 the disparity due to the diversity of design formats and languages prompted outcries from industry where the future was seen as,

> "A nightmare of incompatible formats and a babel of different languages."[12]

The rhetorical question would be, "Has it gotten any better since 1984?" From the surveys of design systems available in the trade press, the answer is no; although efforts by IC designers and fabricators have produced draft interchange formats (e.g. EDIF). In order to couple the large amount of distributed database designs many individual translators have been written. Such one-on-one translation does not provide the integration necessary for automated design and fabrication. Without data integration, no amount of automation will overcome the data interchange problem.

A series of workshops were held to form a base-line for what would constitute the requirements for an EIS. More than 150 people representing near as many organizations attended the workshops. The result was the creation of the DoD Requirements for Engineering Information Systems [13]. Five key areas for an EIS were identified by the participants. An EIS must support:

- the reuse of design information from all forms of input,
- an information repository and data caputre designed for a multi-base, heterogeneous environment,
- an interface to its information model such that it economically supports integration of existing CAE software,
- a system that is not monolithic in use so that installations may tailor the system for current and future needs, and
- the efficiency to support the above functionality in its operation.

The architecture of the EIS is rooted in its information model; which when used, provides a Pax Romana (enforced peace) on the conflict of data representation and data usage. This Information Model is the focus of the EIS effort that will allow the identified key area to be achieved. The requirements are that,

> "The EIS must provide a model of the classes of engineering information that are needed to accurately describe the sem-

> antics of the information in the *engineering environment* in which the EIS operates. The EIS Engineering Information Model (EIM) need not be used to actually represent engineering data; this is the purpose of the common exchange format. Rather, it must provide a definition of all information classes and modeling rules needed as the basis for formulating a conceptual framework for information exhange."[14]

It is this semantic description of engineering information that provides the knowledge-based technology that distinguishes the EIS effort from other data-dictionary based, multi-view databases. It is the goal of the EIM to have the specification of semantics in a precise and understandable form. The information classes and prescribed modeling rules will ensure that the allowable combinations of the data can be modeled in exactly one way; the are no redundant EIM models of the same data within the system.

A goal of the EIS is to develop an accepted *Common Exchange Format* (CEF) in order to promote the exchange of data between design systems, repositories and organizations. From the experience gained in the development of the VHDL, the important factor in data exchange is the information model. Once the model is developed, the development of the exchange format is one of representation notation design. The Object-Oriented Data Language will be used in the EIS for defining the syntax for manipulating objects maintained within an EIS. The PROBE Data Model, an object-oriented extension of DAPLEX, developed by Computer Corporation of America. DAPLEX is a semantic data model and query language that will provide the necessary features for an object-oriented information model; such as

- the concept of an entity or object that has existence independent of its properties or relationships,
- support for relationships between objects and for set-valued properties, and
- types and generalization hierarchies with inheritance.

For access to repositories through the EIS the Object-Oriented Data Language will be used as the CEF between EIS installations. In addition, data exchange adapters will be used to transport design data via VHDL and EDIF. Alternate exchange formats, such as a substantial portion of SQL will be used for non-EIS installations as the program develops thus allowing an interface to foreign information models.

The Object Manager of the EIS is the responsible "agent" for managing objects and functions. It registers new objects, deletes objects that are unneeded, locates and retrieves objects, and provides access to objects. The Object Manager provides services for resolving object references in bindings with application to 1) persistent and temporary objects (data and events), 2) stored and derived objects (database and computed), and 3) passive and active objects (data and processes). Implementation of the Object Manager is based on the design and facilities of the ENCORE system by Brown University.

With the EIS Information Model key-stone set within the EIS, the representation of data is best controlled through rule-processing and control-point activation of data management functions. The *short-term requirements for* rule-based processing of EIM data are that,

> "Rule processing must be supported by programs that implement all required management and control and other rule-based capabilities. There must be an interface specification for every situation in which rule processing is necessary that allows programs to invoke appropriate rule processing programs and pass parameters to them. Rule processing may be implemented via object programs in the short term...." "The EIS must be able to invoke the rule processing services in a heterogeneous, distributed environment. The services must fulfill tool availability requirements..."[15]

In the extended short-term requirements, the general rule-based system which allowed object programs to have static knowledge-bases is modified so that,

> "All rule-based capabilities required by the EIS must be provided by a rule processor, which can be invoked through programs that use the specified" [note: CAIS] "standard interfaces. The rule processor must support the execution of rules specified by a rule spec-

ification language. The EIS must support facilities for adding, deleting, and modifying rules. The rule specification language must support the concept of system supplied variables..." "and must support evaluation of expressions, condition testing and the triggering of actions. The rule specification language must allow for the specification of actions, including sending messages, changing global and object-related management and control information, and invoking programs. The rule specification language must support the concept of variables and parameters. The rule specification language must permit use of any type of object as a variable or parameter and must allow for the specification of parameterized queries containing update operations against EIS-managed data. The EIS, in combination with the rule processor, must be able to support the concept of parameterized messages and programs, and must be able to supply the parameter instantiations automatically." [16]

Thus the EIS Information Model is based on processing information using a multi-rule knowledge base in a multi-base environment. From this foundation the exchange of information among diverse environments is no longer a matter of format, but is one of semantics.

Summary

This paper has covered the descriptive capability and control mechanisms of the VHDL and the Information Model structure of the EIS. It is the purpose of both of these standards efforts to promote the interchange of electronic design data through the semantic content of the data rather than in its physical-/logical format. It is the intent that both of these "tools," a language and an environment, will be platforms from which knowledge based electronics design may continue forward. Internal to the VHDL there exist the necessary control structures and proof mechanisms for the language to be the input to a formal proof of correctness system as done by Dr. Luckham at Stanford University. As has been described above, the EIS Information Model is to be based on known knowledge-base requirements and techniques.

References.

1. Cammarata, Stephanie and Melkanoff, M., "An Interactive Data Dictionary Facility for CAD/CAM Data Bases," Expert Database Systems, Benjamin/Cummings Publishing Co., Menlo Park, CA, 1986, pp. 423-440.

2. King, Roger, "A Database Management System Based on an Object-Oriented Model," Ibid pp. 443-468.

3. Katz, Randal, "Managing the Chip Design Database," IEEE Computer Magazine, Vol.16, No.12, December 1983.

4. Kalay, Y., "A Database Management Approach to CAD/CAM Systems Integration," Proceedings 22nd ACM/IEEE Design Automation Conference, June 1985.

5. Brown, H., C. Tong, Foyster, G., "Palladio: An Exploratory Environment for Circuit Design," IEEE Computer Magazine, Vol.16, No.12, December 1983.

6. Elias, N., Byrne, R., et.al., "The ITT VLSI Design System: CAD Integration in a Multinational Environment," Proceedings 22nd ACM/IEEE Design Automation Conference, June 1985.

7. Preston, G., "Report of IDA Summer Study on Hardware Description Language," HQ 81-23681, Institute for Defense Analyses Science and Technology Division, Arlington, VA, October, 1981.

8. Johannsen, D., McElvain, K., Tsubota, K., "Intelligent Compilation," VLSI Systems Design, April 1987.

9. Janac, George, Carlos, G., Davis, R., "A Knowledge-Based GaAs Design System," VLSI Systems Design, April 1987.

10. Goering, Richard, "Intelligent Silicon Compiler Optimizes ASIC Design," Computer Design, April, 15, 1987.

11. Kemmerer, Richard, "Verification Assessment Study Final Report," C3-CR01-86, Office of Research and Development National Computer Security Center, March 27, 1986.

12. Patton, C. "Languages and Data Formats Vie As Potential Standards in the CAE Design Loop," Electronic Design, Vol.32, No.26, December 27, 1984.

13. Linn, Joseph, Winner, R. editors, "The Department of Defense Requirements for Engineering Information Systems," P-1953, Institute for Defense Analyses, Alexandria, VA, July, 1986.

14. Ibid paragraph 3.24.

15. Ibid paragraph 3.10.

16. Ibid paragraph 4.10.