

The Military Handbook on Documenting Systems with VHDL

Geoffrey A. Frank
Ray C. Anderson
Research Triangle Institute

Gerald Michael
Army Research Laboratory

Abstract

The Army is leading a tri-service effort to develop a handbook on using VHDL to document the design of military digital electronic systems. The handbook is designed to assist its audience in tailoring the VHDL Data Item Description for VHDL contract deliverables so as to achieve the maximum benefit for the government at a reasonable cost. This paper presents an overview of the handbook, discusses four major issues that have arisen in developing the handbook that are still not completely resolved, and presents three recommendations made by the handbook for developers of models that are to be delivered to the government.

1. Introduction

The Army is leading a tri-service effort to develop a handbook on using VHDL to document the design of military digital electronic systems [1]. The intended audience for the handbook includes government personnel involved in system acquisition and document acceptance and verification, as well as defense contractor staff that are planning and implementing the delivery of VHDL descriptions for hardware. The delivery of VHDL models as documentation is required by MIL-STD 454, Requirement 64 [2]. The handbook discusses the requirements in MIL-STD 454 and the VHDL Data Item Description (DID) [3]. The handbook is designed to assist its audience in tailoring the DID for VHDL contract deliverables so as to achieve the maximum benefit for the government at a reasonable cost. It is also intended to help a government agency to evaluate how VHDL will be used to design a system, and to determine how a VHDL model can be evaluated.

2. Overview of the Handbook

The handbook is organized into nine chapters. Chapter 1 is an introduction which discusses the history, purpose, and scope of VHDL; states the purpose, scope, and intended audience of the handbook; and provides a synopsis of the contents of the handbook.

Chapter 2 describes the use of hierarchy in modeling digital electronic hardware and the concepts of behavioral and structural models of electronic systems. This chapter also discusses the uses of abstraction in modeling digital electronic hardware and the use of simulation to support functional correctness checking and performance evaluation.

Chapter 3 discusses basic VHDL constructs from three perspectives:

1. Their ability to capture the structure and behavior of digital electronic systems,
2. Their ability to support reuse and interoperability of VHDL models, and
3. Their ability to support the annotation of VHDL models with descriptive information.

Chapter 4 discusses two government standards which require the use of VHDL:

- MIL-STD-454 [2],
- the VHDL DID [3], DI-EGDS-80811.

This chapter provides guidelines for tailoring the VHDL DID and also describes required annotations

for VHDL models.

Chapter 5 contains a description of the construction and use of behavioral VHDL models. Common techniques for creation of behavioral VHDL models, timing specifications for behavioral models, and annotation of behavioral models are described. Also discussed are the usefulness of behavioral models in top-down design and simulation of models with mixed levels of abstraction.

Chapter 6 discusses the construction and use of structural VHDL models. Common techniques for the creating of structural VHDL models, including automatic synthesis and schematic capture are described. Applications of structural models for hybrid model simulation, physical design, testability analysis, and annotation with layout and testability information are described.

Chapter 7 describes the preparation of VHDL models for simulation. This chapter includes an overview of WAVES and comparisons of VITAL and EIA 567. The process of configuring a model from libraries of component descriptions is described. This chapter emphasizes techniques that help to support the interoperability of models. This chapter also compares the parameterized timing models of VITAL and EIA 567.

Chapter 8 discusses issues related to the modeling of test and diagnostic functions of digital electronic systems with VHDL. This chapter describes measures and techniques for testability and describes different levels of testability based on the IEEE 1149 hierarchy of testing interfaces.

Chapter 9 describes the preparation of a VHDL model for delivery to the government. The contents and organization of the files that are delivered to the government, as specified in the VHDL DID, are described. The files that must be delivered include not only the VHDL source models, but also test vectors, annotations, and other external files, as well as other documentation. Chapter 9 also includes recommendations for VHDL model style, as well as recommendations for naming files and organizing libraries.

The handbook also features an extensive glossary, a list of acronyms and abbreviations, and an extensive index. There are two appendices: a copy of the VHDL model verification procedures developed by Rome Laboratory [4], and a copy of the VHDL DID as tailored for the Rapid Prototyping of Application Specific Signal Processors (RASSP) program's prime contractors [5]. Originally, a complete end-to-end example had been planned as an appendix. The work of the Navy's Technology Independent Representation of Electronic Products (TIREP) project [6] serves this purpose and is extensively referenced in the handbook.

3. Issues Raised by the DID

Planning how to respond to the DID requirements raises several major issues that are discussed in the handbook. These issues raise both technical and contractual issues that contracting officers and contractors will have to resolve as they tailor the DID to the needs of specific contracts. The basic drivers for these negotiations are cost on the one hand versus the utility of the models delivered.

3.1 Level of Abstraction

A key issue that a contract monitor and a contractor must negotiate is what level of abstraction is required for models delivered to the government in accordance with the DID. In general, the cost of models at lower levels of abstraction are greater than the cost of models at the higher level. Models at lower levels of abstraction are easier to reverse engineer, and they may reveal more about the implementation of a proprietary design than the contractor wants. At the same time, higher level models may be more implementation independent and therefore more useful as specifications for the redesign of obsolete parts in new technologies.

The DID distinguishes between two levels of abstraction of models: behavioral and structural.

Chapter 2 of the handbook describes the differences in these models as defined in the DID. The DID requires that a behavioral model describe both function and timing in an implementation independent fashion. The DID requires that a structural model must provide the level of detail required for fault coverage analysis. The DID also encourages the interoperability of structural and behavioral models of components so that models with mixed levels of abstraction can be configured.

In order to give the user some options in interpreting the DID requirements for behavioral and structural models, the handbook describes five levels of abstraction:

- **Network Models.** These models are also known as Processor Memory Switch (PMS) models [7]. The handbook describes two special cases of network models: performance models and interface models. Performance models may provide only timing information and not simulate the functions of the system. Interface models combine high level and incomplete models of the processor and memory components with detailed and complete behavioral models of bus and network interfaces. In general, network models are not fully functional; they are used to verify specific characteristics of a design. They do not meet the requirements that the DID states for behavioral models. Despite this deficit, they have played a key role in system development. For example, interface models were used to verify the interoperability of components developed by different contractors for the Air Force's F-22 program. The development of performance models are a key component of the RASSP prime contractor design processes [8,9].
- **Algorithmic Models.** These models are the highest level behavioral models as required by the DID. However, the data structures for the signals connecting components in an algorithmic model are abstract, so that it may not be possible to use these models in an interoperable and portable manner. For example, signals of the floating point type cannot be interfaced to a component with bit vector ports since the representation of floating point numbers is implementation dependent in VHDL.
- **Instruction Set Architecture Models.** An ISA model accurately describes all the functions and data types provided by the hardware that are accessible to the hardware user. In particular, a correct ISA model of a programmable device will correctly execute any valid program for the device. These models are critical to virtual prototyping [10]. They are a sufficiently detailed representation of the hardware to permit the execution of compiled software. This is an important test and integration function in a software/hardware codesign process. Since these models require bit accurate interfaces, they do meet the interoperability requirements stated in the DID. If an ISA model includes both the complete functionality and the timing of the hardware being modeled, then it fully satisfies the requirements of the DID for a behavioral model. The concept of an instruction set architecture is usually applied to a CPU, but can be extended to include any hardware component that has any switches or parameters that are set in the field.
- **Register Transfer Level Models.** A register transfer level model describes the functions and data types accessible to the user of the hardware, and also includes descriptions of the internal memory (or registers) and the internal data paths of the hardware. Register transfer level models are an important class of models because hardware synthesis technology is reaching the point where it can be used to generate detailed integrated circuit designs from register transfer level models.
- **Gate Level Models.** These models can be used to satisfy the fault coverage analysis requirements stated in the DID for structural models. Therefore, they are essential for satisfying the DID requirement for structural models.

3.2 Configuration of VHDL Models

The DID requires that VHDL models be structured hierarchically and that the hierarchical partitioning of the system follow the physical partitioning of the hardware. The DID requires both behavioral and structural models for all components in the hierarchy except approved leaf level modules. The intent of the DID is to support the construction of models of mixed levels of abstraction where some components are represented by abstract behavioral models and some components are represented by detailed structural models. Implementing a VHDL design database

that serves this intent imposes several technical challenges:

- The hierarchy of components needs to have a corresponding hierarchy of definitions for signals. For example, a bus may be represented in an abstract model as a single signal that is defined as a record data type. The same bus may be viewed as a collection of separate bit signals by a register transfer level model of the same component. The use of type conversion functions in port map statements applies only when there is a one-to-one mapping between the ports of the component and the corresponding signals. In this case, the signal has many corresponding ports. A wrapper entity can be used to separate the elements of a composite signal and to connect them to the appropriate ports. The handbook recommends the use of composite data types for signals in structural models where that is appropriate.
- A typical VHDL model of a design will be built with an extensive set of packages to implement associated standards, such as IEEE 1164 [11], VITAL[12], and EIA-567[13]. How these packages should be integrated into a model at a mixed level of abstraction is not clear. For example, the EIA 567 timing view package specifies timing information for individual pins. How does a high level model which uses record types for signals and component ports be made compatible with EIA 567? As another example, WAVES level II [14] supports enumerated types, but can it be used effectively with record type ports on a MUT?
- An alternative to the use of wrappers as described above is the use of alternative data type definitions encapsulated in packages. When the level of abstraction is changed, most of the text of the high level structural models remains the same, but the packages that are referenced change. These changes require editing the library and/or the use statements of the reanalysis of the entities and architecture bodies that use the packages. The alternative package definitions must either be stored in different libraries or be named differently. Therefore changing the level of abstraction of a component in a structural architecture would require editing the architecture body.
- In order to deliver a VHDL design database that supports the configuration of models with mixed levels of abstraction, a framework is required. Typically, this framework is constructed as a set of "soft standards" implemented with a collection of VHDL packages that are referenced by all the design entities in the database. Current standardization efforts are proceeding bottom-up, capturing what has become accepted practice in the industry. Standards for high-level models and models with mixed levels of abstraction do not yet exist. The handbook discusses relevant existing standards and identifies issues that need to be resolved as part of DID-related negotiations. During these negotiations, the capabilities of the framework that will be delivered should be made clear for the benefit of both the contracting agent and the contractor. The specification and implementation of such a framework remains a major technical challenge. What part of the VHDL description of a performance model should be held in common with a more detailed behavioral or structural model of the same hardware? What "soft standards" can be defined that will allow back annotation of simulation and analysis of detailed behavioral models into higher level performance models?

3.3 Modeling of Built In Test Functions

The VHDL DID levies specific requirements for the modeling of testability in both the behavioral and structural models. The requirements for implementation-independent behavioral models that deal with Built In Test (BIT) functions seems contradictory.

3.3.1 Modeling BIT in Behavioral Models

The VHDL DID requires that "Test and maintenance functions which are part of the physical unit and are available to the user shall be included in the behavioral body." However the VHDL DID also states that "signal values which are dependent upon a particular structural implementation, such as scan path signatures, shall not be specified in the behavioral model." The handbook proposes a modeling strategy for meeting both of these requirements, which may at first seem to be contradictory. The crux of the issue is to what extent can test and maintenance features be described in a manner that is implementation independent, since the goal of a VHDL behavioral

model is to provide the user with a model that is free of implementation dependencies and can be used as the specification for multiple implementations.

The handbook recommends that VHDL behavioral models should be developed so that they are able to verify that when hardware modules are properly interconnected, their test controllers, test buses and test interfaces are able to communicate effectively. For example, test buses such as the IEEE 1149.1 [15] and 1149.5 [16] define a set of instructions that can be communicated through those buses. Behavioral models of modules connected to these buses should be able to interpret those instructions and respond appropriately. However, the DID does not require that behavioral models produce the same results as the actual hardware when diagnostic functions are executed. Behavioral models of the IEEE 1149.1 TAP controller exist [17], and models are being developed for the backplane test bus [16]. Both of these bus structures define a set of test instructions supported by the bus and its controllers.

The handbook recommends that VHDL behavioral models and their test benches should be able to evaluate different strategies for interconnecting the test subsystems proposed in a hardware design. If the models of the test and maintenance buses and their interfaces include timing information about the buses, then the VHDL behavioral model can be used to explore the test times for different configurations of modules and test buses.

3.3.1 Modeling BIT in Structural Models

The test and maintenance assessment issues are a critical part of the specification of how detailed a VHDL structural model must be in order to meet VHDL DID requirements. The VHDL DID requires that "Structural bodies shall represent the physical implementation accurately enough to permit logic fault modeling and test vector generation. Structure which is created to support testing and maintenance such as scan paths shall be included in the VHDL structural description." Logic fault modeling typically requires modeling the effects of stuck-at-zero and stuck-at-one faults on all logic level interconnects in a logic-level model of the system. The use of higher level models to estimate fault coverage are not as accurate [18].

3.4 Choice of Terminology

Because a DID is a contractual document, it is important that both the contracting agency and the contractor have a common understanding of the terms used in it. This is particularly important in the VHDL DID since this is a very technical specification of deliverables. At a more global level, reaching consensus on the terminology used in the DID or in tailoring of the DID is essential as a preliminary step towards the creation of standards for higher level models. These standards are essential if the DoD is to be able to use high level VHDL models as executable specifications for the reengineering of obsolete parts.

Review of drafts of the handbook revealed a much greater disparity in interpretation of terminology than the authors had anticipated. We anticipate that the glossary of the handbook can serve as a "stake in the ground" for defining terminology.

An initial problem arises in the use of the terms "structural" and "behavioral" in the DID. The concept of structural and behavioral architecture bodies is defined syntactically in the 1987 version of the LRM [19]. The 1994 version of the language does not support the STRUCTURE and BEHAVIOR attributes of architecture bodies [20]. As described previously, the DID also defines what it means by structural and behavioral models. Although these two definitions are not inconsistent, they are by no means synonymous. The DID places specific semantic requirements on behavioral (e.g., the inclusion of both function and timing) and structural models (e.g., the support of logic level fault coverage analysis) that are not addressed by the VHDL-1987 syntactic definitions. The definitions of behavioral and structural models are further confused by the common use of behavioral model as a model at some higher level of abstraction than register transfer

and structural model to refer to a logic level model. The handbook highlights the very specific technical and contractual use of behavioral and structural model stated in the DID.

Discussion of the terminology associated with levels of abstraction has fueled two Birds of a Feather sessions; one at the last VIUF, and one at the RASSP Principal Investigators conference. The RASSP educator/facilitator is now circulating a position paper describing a more formal model of levels of abstraction [21]. One term in common use that we would like to see eliminated is "bus functional model". We prefer the use of the term interface model.

4. A Summary of Recommendations for Tailoring of the DID

The handbook establishes many guidelines on preparing VHDL models for delivery to the Government. Three of the high level guidelines are discussed in the following paragraphs.

4.1 Define Your Modules

The DID uses the term module in a specific contractual sense. Modules are the basic deliverable items described in the VHDL DID. Each module consists of a behavioral model, a structural model (except for leaf level modules, which are also specifically defined in the DID), a test bench, and any supporting documentation or external data files that must be delivered for each module. Careful definition of the boundaries of modules can reduce the level of effort required without sacrificing the quality of the model. For example, the cost of developing a test bench can equal the cost of developing a model of the hardware component that is tested by the test bench. A simple interpretation of the DID would consider each design entity in the system being documented as a separate module. However, defining modules to consist of several design entities can reduce dramatically the number of test benches that have to be created, thus reducing the cost of building the VHDL models.

According to the DID, a complete VHDL model consists of a hierarchy of modules. The hierarchy of modules must follow the physical design hierarchy of the system being modeled. When the modules are defined, certain boundaries should be preserved, such as Line Replaceable Units (LRUs). Any preplanned product improvements should also be considered in defining these boundaries. From the Government point of view, the executable nature of VHDL models makes them valuable as specifications for Form, Fit, Function upgrades or replacement of obsolete parts. This value is enhanced by the requirement that test benches be provided with the VHDL models. This value is negated if the part that is to be redesigned or replaced does not correspond to one of the modules that was delivered when the part was initially built.

4.2 Define Your Design Process In Terms of Model Delivery

If VHDL modeling is to be used to assist in the design process, as opposed to just documenting the results of the design effort, then the role of VHDL models in the design process should be defined. This definition of the design process should specify the delivery of VHDL models as key milestones in the system development process. For each delivery, the level of abstraction of the models, the scope of the models, and the purpose of the models (as implemented by the accompanying test benches) should be specified. This approach to the definition of a design process is similar to the MIL-STD 2167 approach, but uses executable models instead of paper documents. The executable nature of VHDL models makes them much more valuable than paper documents. However, it creates an additional burden on the contracting agency to ensure the validation/verification process takes advantage of this nature. A validation /verification process can include simulation of the models if the test benches are closely linked with the hardware and system test plans. The integration of test benches into test plans is a major area of research that can yield valuable results in terms of ensuring that the effort required to develop VHDL models pays off in reducing design errors [22].

4.3 Define Soft Standards for Behavioral Modeling

Soft standards are needed for high-level behavioral models that serve the purpose which standards such as IEEE 1164 [11], WAVES [14], EIA 567 [13], and VITAL [12] serve for register-transfer level and gate-level models. These soft standards should define the data structures used for signals in behavioral models, provide structures for high level test benches, and provide formats for high-level timing information such as instruction set times. These soft standards should be formalized through the use of VHDL packages. IEEE 1164 [11] is a good example of a standard that defines an abstract data type with a VHDL package. The package contains a data type definition, definitions of associated constants, a resolution function, and type conversion functions for related data types. EIA 567 [13] describes timing views, electrical views, and physical views for VLSI circuits. The Honeywell performance models [23] provide a high-level timing view in the form of packages of instruction timing information similar to the timing information in the data sheet for a commercial microprocessor. The format of packages like these need to be standardized so that libraries of data sheet information can be created, validated, and reused. The Honeywell models also provide a package defining a token data type and some associated functions. These high level data types also need to be standardized.

5. Conclusions

Based on the reactions of a wide range of reviewers of draft forms of the handbook, we believe that the handbook will serve as a valuable aid to those negotiating complex contractual and technical issues related to the delivery of VHDL models as documentation for electronic hardware designs. Writing this handbook has uncovered many open research issues and has revealed a broad and fervent diversity of opinions on terminology, development style, and interpretation of the DID. We look forward to helping to build a consensus on many of these issues that can lead to the standards that are needed if VHDL is to achieve its potential for saving the Government money during systems design.

Publication of the handbook is planned for October of 1995. We are interested in broader forms of dissemination of the information in the handbook. We are pursuing the publication of our approaches and recommendations concerning these issues in the form of a reference textbook by a commercial publisher. We are also interested in publication of the handbook in a hypertext format.

References

1. Army Material Command. *Documentation of Digital Electronic Systems with VHDL*. Draft Handbook, 31 December 1994.
2. MIL-STD-454M. *Standard General Requirements for Electronic Equipment*. Department of Defense, 15 December 1989.
3. DI-EGDS-80811. *VHSIC Hardware Description Language (VHDL) Documentation*. Department of Defense, 11 May 1989.
4. Rome Laboratories/ERDD. *VHDL Model Verification and Acceptance Procedure*. Technical report, Department of the Air Force, Griffiss Air Force Base, Rome, NY, March 1992.
5. ARPA/ESTO. "VHDL Technical Reports". *Contract Data Requirements List*, Contract Line Item 0009AA, Exhibit D, Contract DAAL01-93-C-3380, Martin-Marietta. 21 July 1992.
6. Naval Research Laboratory (NRL), the Naval Surface Warfare Center (NSWC) and the Naval Air Warfare Center, Aircraft Division (NAWC-AD). *A VHDL Modeling Guide*, Technical Report of the TIREP project, May 1994.
7. D. P. Siewiorek, C.G. Bell, and A. Newell. *Computer Structures: Principles and Examples*. McGraw-Hill Book Co., Inc., New York, NY, 1982.
8. J. Pridmore and W. Schaming. "RASSP Methodology Overview". in *Proceedings of the 1st Annual RASSP Conference*, Arlington VA, August 1994.
9. R. Dreiling. "Processes and Experiences in VHDL Top-Down Design". in *Proceedings of the 1st*

- Annual RASSP Conference*, Arlington VA, August 1994.
10. M. Richards. "The RASSP Program: Overview and Accomplishments". in *Proceedings of the 1st Annual RASSP Conference*, Arlington VA, August 1994.
 11. IEEE Std 1164-1991. *IEEE Standard Logic Package*. The Institute of Electrical and Electronic Engineers, New York, NY, October 1991.
 12. VITAL Steering Group. *VITAL: VHDL Initiative Towards ASIC Libraries*. Draft Standard, Version 2.2a, March 1994.
 13. Electronic Industries Association. *VHDL Hardware Component Modeling and Interface Standard*. EIA 567-B, Washington, DC, March 1994.
 14. IEEE Std 1029.1-1991. *Waveform and Vector Exchange Specification*. IEEE Design Automation Standards Subcommittee, New York, NY, 1991.
 15. IEEE Std 1149.1-1990. *IEEE Standard Test Access Port and Boundary-Scan Architecture*. IEEE Test Technology Technical Committee, New York, NY, 21 May 1990.
 16. P. McHugh. "IEEE P1149.5 Module Test and Maintenance Bus". *IEEE Design and Test of Computers*, 1992.
 17. P. M. Campbell, M. Vai, and Z. Navabi. "Implementation of IEEE Std. 1149.1-1990 in VHDL". In *Using VHDL in System Design, Test, and Manufacturing*, 1992. VHDL International Users' Forum, Scottsdale, AZ.
 18. McGough, J. G., F. L. Swern, S. J. Bavuso, "New Results in Fault Latency Modeling". in *Proceedings of the IEEE EASCON*, Washington, D.C., September, 1983.
 19. IEEE STD 1076-1987. *IEEE Standard VHDL Language Reference Manual*. The Institute of Electrical and Electronic Engineers, New York, NY, March 1988.
 20. IEEE STD 1076-1993. *IEEE Standard VHDL Language Reference Manual*. The Institute of Electrical and Electronic Engineers, New York, NY, 1994.
 21. Madiseti, Vijay, *A Taxonomy for VHDL Modeling in RASSP DSP Laboratory Technical Report 94-14*, Georgia Tech-ECE, 20 December 1994.
 22. J. R. Armstrong, G. A. Frank, Hrishi, Prabahacar, Z. Xu," Behavioral Testbench Generation". in *Proceedings of the VHDL International Users' Forum Spring Conference*, VHDL International Users' Forum, San Diego, CA, April 1995..
 23. F. Rose, T. Steeves, T. Carpenter. "VHDL Performance Modeling". in *Proceedings of the 1st Annual RASSP Conference*, Arlington VA, August 1994.