Instructor’s Manual

RASSP E&F VHDL Modules

Modules 10-13

RASSP Education & Facilitation Program

See the RASSP Disclaimer file for additional RASSP Disclaimer, Warranty and Limitation of Liability Information concerning the material, VHDL code and software developed under the RASSP programs or incorporated in RASSP material.

Table of Contents

1 Definition of Topic

This instructor's manual provides additional information for the RASSP E&F VHDL Modules, M10 through M13. These modules provide an introduction to VHDL so that the examples found in most of the other RASSP E&F educational modules may be readily understood. In addition, the student will gain a good foundation in the understanding of VHDL semantics, syntax, constructs, and structure.

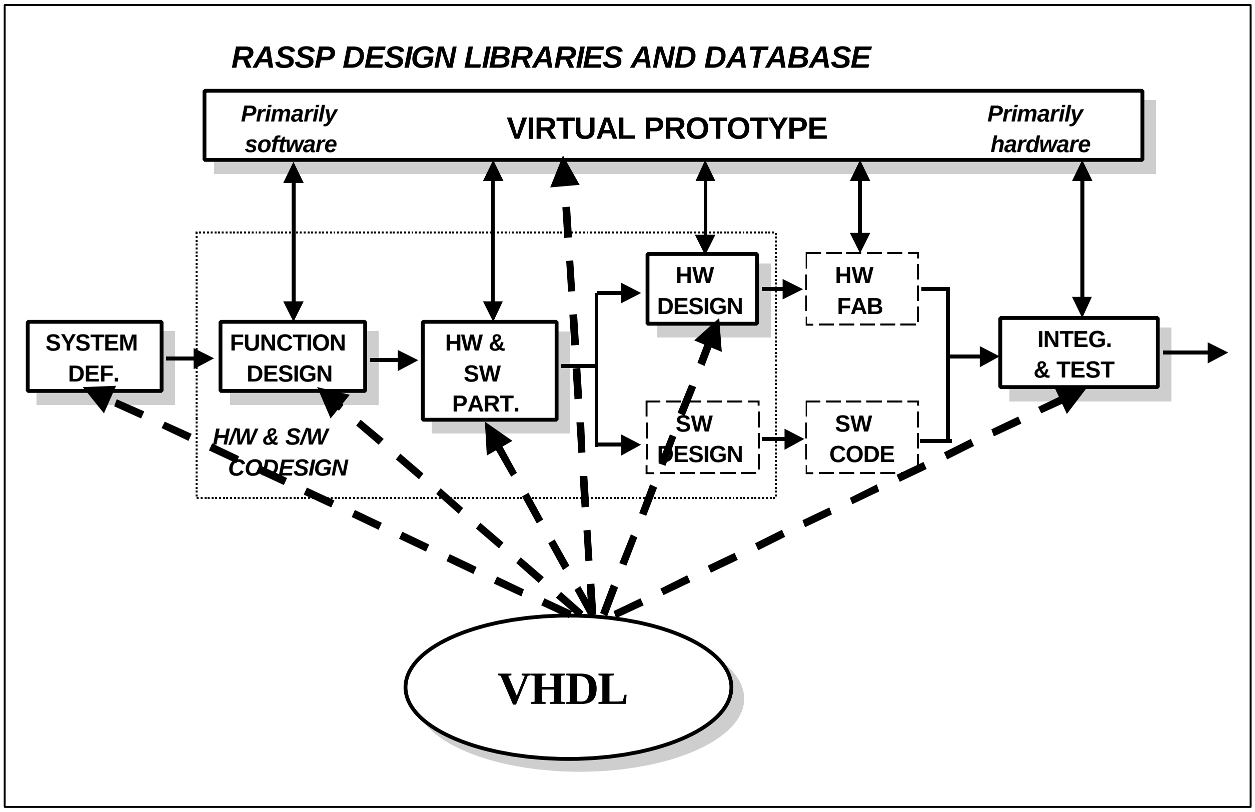

Figure 1. VHDL in RASSP

2 Rationale

VHDL (i.e. **V**ery High Speed Integrated Circuit **H**ardware **D**escription **L**anguage) is a powerful and popular digital electronic system description language. VHDL has been approved as a standard by both the Institute of Electrical and Electronic Engineers (IEEE) and the U.S. Department of Defense for the description of digital electronic systems. VHDL is able to describe systems at various levels of design abstraction from block-oriented system level descriptions to gate-level digital circuit descriptions. As such, VHDL supports a number of RASSP concepts particularly well. For example, virtual prototypes described in VHDL can be designed to reflect the ultimate system implementation very accurately.

Although this series of instructional modules was originally developed to enable the student who may have had little or no experience with VHDL to understand the examples that are found in the various other RASSP E&F educational modules, the VHDL modules can be used to provide a good introduction to the language to graduate and advanced undergraduate students as well as engineers involved in the design of digital electronic systems.

2.1 Related Modules

A number of RASSP E&F modules provide examples described using VHDL. Two modules which are heavily dependent on VHDL, however, and may provide valuable information in the use of the language in actual applications are Module 32, *Virtual Prototyping Using VHDL*, and Module 60, *Hardware Synthesis Using VHDL*.

3 Module Content Description

3.1 Basic VHDL

Module 10, *Basic VHDL*, provides an introduction to the VHSIC Hardware Description Language and its fundamental concepts. VHDL is a language specifically developed to describe digital electronic hardware and its attributes. VHDL is a flexible language and can be applied to many different design situations. This language has several key advantages, including technology independence and a standard language for communication. The module describes many of the advantages of using VHDL and a short history of the language.

As an introduction to the language, a small design example is shown. This example shows three different methods of describing the hardware. The module then introduces three key models of hardware in VHDL: behavior, structure, and time. These models are important aspects of VHDL and must thoroughly understood. Several basic VHDL constructs are shown next. The module describes the basic data types and objects available to the designer. Built-in data types include integers, floating point numbers, and many others. The designer also has variable, signal, and constant objects available. The difference between sequential and concurrent statements is also explained. The basic VHDL design units, the entity and the architecture, are described in detail. Furthermore, VHDL supports code reuse through its packages and libraries. Finally, the basic operators and attributes of the data types and objects are shown.

Module 10 concludes with a small but comprehensive design example to illustrate many of the constructs introduced in the earlier sections of the module. Large examples are provided in the later modules. The focus of this module is to introduce the basic features of this language and to provide brief examples of its use.

3.2 Structural VHDL

Module 11, *Structural VHDL* , describes the use of VHDL for describing models in terms of component instantiations and interconnections. *Structural VHDL* can be appropriate at any level of design. For example, testbenches for completed components are often described using structural VHDL. Furthermore, structural VHDL supports the use of libraries and component reuse. The various mechanisms available in VHDL to incorporated predefined components into a VHDL description are presented. The use of idealized local ’components’ is described as well as generate statement to create regular structures easily (e.g. RAM, ROM). Additionally, this module shows how VHDL supports libraries and component reuse. Components in structural VHDL are fully described outside the architecture, most often in component libraries. Configuration of these components involves selecting an entity and architecture for the component and specifying parameters for the component. The module concludes with a comprehensive example to illustrate much of the material covered in the early sections.

3.3 Behavioral VHDL

Module 12, *Behavioral VHDL*, describes features of the language that describe the behavior of components in response to signals. Behavioral descriptions of hardware utilize software engineering practices and constructs to achieve a functional model. Timing information is not necessary in a behavioral description, although such information may be included easily.

The VHDL process construct is described first. Processes run code sequentially. The statements allowed in a process, referred to as ’sequential’ statements, are listed in the module. Subproerams are another behavioral construct; they support code reuse description simplification. One use of subprograms is in bus resolution functions which allow the description of buses with multiple signal drivers. A discussion of packages is also included. Packages can contain subprogram descriptions, custom data types, and numerous other VHDL model fragments that a designer may wish to reuse easily. Finally, the module describes the use of testbenches and lists some problems to avoid in VHDL.

The *Behavioral VHDL* module concludes with a comprehensive example using the quicksort routine. Although a detailed understanding of the algorithm implemented by this routine are not important for a full understanding of the VHDL constructs presented in this module, the example serves as a vehicle for highlighting many of the VHDL features presented in this module. The model also illustrates the similarity between process-oriented VHDL descriptions and other general-purpose high-level programming languages.

3.4 Advanced Concepts in VHDL

Module 13, *Advanced Concepts in VHDL*, spans a wide range of topics, including several that may be applied to higher levels of design abstraction.. Many of these constructs will have been introduced in the first three VHDL modules in this sequence, but this module covers them more comprehensively. Examples of such constructs include signal assignment statements, and the capabilities and differences when they are used as concurrent VHDL statements or sequential

VHDL statements. Similarly, the VHDL process is discussed in more detail than in earlier modules. It should also be noted that TEXTIO and shared variables are introduced in this module.

Two illustrative examples showing the use of many of the constructs discussed are provided. The first example shows the implementation of a simple Abstract Data Type implemented with custom VHDL data type and subtype declarations and overloading operators to manipulate signals and variables of those VHDL types. The second example, based on the University of Virginia's ADEPT system, shows the use of records and subprograms (including a Bus Resolution Function) to implement a simple performance-oriented model using VHDL.

4 Teaching Options

The series of modules begins with Module 10, *Basic VHDL*, providing an introduction to the basic structure of VHDL modules and the VHDL simulation cycle. This module provides an essential starting point for students who are not familiar with VHDL. The module includes some simple, yet complete, examples of VHDL code to illustrate some of the fundamental concepts.

Subsequent to Module 10, the instructor has the option of presenting Module 11 or Module 12 in either order. Whereas both *Structural VHDL* and *Behavioral VHDL* require an understanding of the material covered in Module 10, the presentation materials in these two modules are relatively independent of each other. In fact, certain instructors may skip one or the other of these two modules, depending on their particular objectives. Module 11, *Structural VHDL*, may be adequate for students who will be developing models by interconnecting existing components provided in prescribed VHDL libraries. Module 12, *Behavioral VHDL*, may be adequate for students who will be developing high-level abstract system models with very shallow design hierarchies.

It must be noted that the laboratory exercises provided with these VHDL modules does follow a sequence starting with M10 and proceeding in numerical order to M13. For this reason, it is recommended that instructors who will be presenting both of the modules present Module 11 before presenting Module 12. Since the complete VHDL source code is provided, instructors who only wish to provide one of these two modules, however, can still make use of the appropriate lab exercises for that module.

Module 13, *Advanced Concepts in VHDL*, is intended to be the last module presented in this sequence, particularly in cases where all the modules will be presented. Much of the material provided in this module has already been introduced in the prior three modules, but at a more basic level. This module revisits many of the concepts and constructs that should already be familiar to the student, but provides a more complete description of the available syntactic forms and semantics of those constructs.

Annotated Bibliography

The texts listed below are readily available and cover the constructs, syntax, and semantics of the VHDL language. Many of these texts also provide complete examples which may be useful to a student who is new to the language:

[1] [Ashenden], Ashenden, Peter J., The Designer’s Guide to VHDL, Morgan Kaumfmann, 1996.

[2] [Ashenden], Ashenden, Peter J., “The VHDL Cookbook,” available via ftp from thor.ece.uc.edu.

[3] [Bhasker95] Bhasker, J. A VHDL Primer, Prentice Hall, 1995.

[4] [Coelho89] Coelho, D. R., The VHDL Handbook, Kluwer Academic Publishers, 1989.

[5] [Lipsett89] Lipsett, R., C. Schaefer, C. Ussery, VHDL: Hardware Description and Design, Kluwer Academic Publishers, , 1989.

[6] [Navabi93] Navabi, Z., VHDL: Analysis and Modeling of Digital Systems, McGraw-Hill, 1993.

[7] [Perry94] Perry, D. L., VHDL, McGraw-Hill, 1994.

The definitive word on VHDL is provided in the Language Reference Manual published by the IEEE. This manual is very useful as a reference to accompany any of the texts above, but is somewhat difficult to read as a standalone text.

[8] [IEEE93] IEEE Standard VHDL Language Reference Manual, IEEE Std 1076-1993