

VHDL Introduction

2 days

70% Lecture, 30% Lab

Basic Level

Overview

This two-day fast-track primer course in VHDL is recommended for the broader corporate audience including application engineers, R&D personnel, hotline engineers, technical writers, and engineering managers. People whose role demands the ability to read VHDL code, understand VHDL-specific issues, identify inefficient or non-synthesizable constructs, and support those who write code will benefit from this course. Both synthesis and simulation aspects of the language are covered. Practical lab exercises provide course participants with direct insight into how the capabilities of the language are used to develop and verify complex ICs, using today's EDA tools.

The course may be customized for company specific topics and areas of focus.

Benefits

Upon completion of this course, students will:

- develop high-level behavioral models to capture design specifications
- refine models to structural detailed designs
- use simulation tools to test and debug models
- identify coding styles that enhance correctness and maintainability of models

Intended Audience

This course is recommended for people with little or no knowledge of VHDL who need to gain a working understanding of the language basics. It is for people who need to understand the VHDL language but will not be writing VHDL code. For people who need to learn VHDL and will be writing VHDL code we recommend the four day class:

VHDL for Hardware Designers

Prerequisites

Students need to be familiar with the basics of digital design such as shift registers, adders, multiplexors and finite state machines.

Follow-on course:

Advanced VHDL Coding Styles for Synthesis & Verification

Training Approach

This is an intensive, interactive course, which is approximately 70% lecture and 30% lab. Questions are highly encouraged.

Course Outline

Day 1

Introduction to VHDL modeling concepts

- uses of modeling in the design flow
- basic VHDL concepts through examples
- lexical elements

Scalar data types and operation

- constants and variables
- predefined types and operations
- subtypes
- predefined subtypes
- user-defined types
- integer types
- enumeration types
- expressions and operators

Sequential statements

- if statements
- case statements
- loop and exit statements
- while loops
- for loops
- assert statements

Composite data types and operations

- arrays, aggregates and array attributes
- array indexing, slicing and operations
- unconstrained array types
- predefined array types
- unconstrained array ports

Lab: Pulse-Width Modulator

Behavioral modeling

- review of entities and architectures
- signal assignment with delay
- attributes of signals
- wait statements

- delta delays
- inertial delay
- concurrent assignments

Structural modeling

- direct entity instantiation and port maps
- positional and named association
- association with expression

Design processing

- libraries
- library and use clauses
- analysis order
- elaboration

Lab: ALU for the eLucid-8 Microcontroller

Day 2

Subprograms

- procedures & return statements
- procedure parameters
- default parameter values
- unconstrained array parameters
- concurrent procedure call statements
- functions
- modeling using functions
- the now function
- overloading subprograms and operator symbols

Packages

- package declarations
- use clauses
- subprograms in packages
- package bodies
- predefined packages

Lab: Packages for Timing Checks and Test Utilities

Aliases

- data object alias declarations
- use with unconstrained arrays

Resolved signals

- resolved subtypes and signals
- composite/resolved types
- std_logic and std_logic_vector
- resolved ports
- type conversions in port maps

Generics

For more information contact:

Tom Wille

tw@tm-associates.com

503-656-4457

TM Associates, Inc.

www.tm-associates.com

- generic lists and generic maps
- generics for delay parameters
- generics for port sizing

Conclusion

- overview of additional VHDL features
- further resources

Lab: Parallel I/O controller for the eLucid-8
Microcontroller