AMD
together we advance_

Designing with VHDL

LANG-VHDL (v1.0)

Course Specification

Course Description

This course provides a thorough introduction to the VHDL language.

The emphasis is on:

- Writing efficient hardware designs
- Performing high-level HDL simulations
- Employing structural, register transfer level (RTL), and behavioral coding styles
- Targeting AMD devices specifically and FPGA devices in general
- Utilizing best coding practices

What's New for 2023.1

- Added new module: Vivado™ Simulator Good Coding Practices
- All labs have been updated to the latest software versions

Level – FPGA 1

Course Details

- 3 days ILT
  Course Part Number – LANG-VHDL

Who Should Attend? – Engineers who want to use VHDL effectively for modeling, design, and synthesis of digital designs

Prerequisites

- Basic digital design concepts
  - Flip-flops and logic gates
  - Basic understanding of synchronous designs

Software Tools

- Vivado Design Suite 2023.1

Hardware

- Architecture: N/A*
- Demo board: Zynq™ UltraScale+™ MPSoC ZCU104 board*

* This course does not focus on any particular architecture. Check with your local Authorized Training Provider for the specifics of the in-class lab board or other customizations.

After completing this comprehensive training, you will have the necessary skills to:

- Implement the VHDL portion of coding for synthesis
- Identify the differences between behavioral and structural coding styles
- Distinguish coding for synthesis versus coding for simulation
- Use scalar and composite data types to represent information
- Use concurrent and sequential control structure to regulate information flow
- Implement common VHDL constructs (finite state machines [FSMs], RAM/ROM data structures)
- Simulate a basic VHDL design
- Write a VHDL testbench and identify simulation-only constructs
- Identify and implement coding best practices
- Optimize VHDL code to target specific silicon resources within AMD FPGAs and adaptive SoCs
- Create and manage designs within the Vivado Design Suite environment

Course Outline

Day 1

- **Introduction to VHDL**
  Discusses the history of the VHDL language and provides an overview of the different features of VHDL. {Lecture}
- **VHDL Design Units**
  Provides an overview of typical VHDL code, covering design units such as libraries, packages, entities, architectures, and configuration. {Lecture, Lab}
- **VHDL Objects, Keywords, Identifiers**
  Discusses the data objects that are available in the VHDL language as well as keywords and identifiers. {Lecture}
- **Scalar Data Types**
  Covers both intrinsic and commonly used data types. {Lecture}
- **Composite Data Types**
  Covers composite data types (arrays and records). {Lecture}
- **VHDL Operators**
  Reviews all VHDL operator types. {Lecture}
- **Concurrency in VHDL**
  Describes concurrent statements and how signals help in achieving concurrency. {Lecture}
- **Concurrent Assignments**
  Covers both conditional and unconditional assignments. {Lecture, Lab}
- **Processes and Variables**
  Introduces sequential programming techniques for a concurrent language. Variables are also discussed. {Lecture, Demo, Lab}

Day 2

- **Control Structures in VHDL: if/else, case**
  Describes conditional statements such as if/else and case statements. {Lecture, Lab}
- **Sequential Looping Statements**
  Introduces the concept of looping in both the simulation and synthesis environments. {Lecture, Lab}
- **Delays in VHDL: wait Statement**
  Covers the wait statement and how it controls the execution of the process statement. {Lecture}
- **Introduction to the VHDL Testbench**
  Introduces the concept of the VHDL testbench to verify the functionality of a design. {Lecture, Lab}
- **VHDL Assert Statements**
  Describes the concept of VHDL assertions. {Lecture}
- **VHDL Attributes**
  Describes attributes, both predefined and user defined. {Lecture}
- **Vivado Simulator Good Coding Practices (VHDL)**
  Discusses the simulation tool and its typical capabilities. {Lecture}
- **VHDL Subprograms**
  Covers the use of subprograms in verification and RTL code to model functional blocks. {Lecture}
- **VHDL Functions**
  Describes functions, which are integral to reusable and maintainable code. {Lecture, Lab}

- **VHDL Procedures**
  Describes procedures, common constructs that are also important for reusing and maintaining code. {Lecture}

Day 3

- **VHDL Libraries and Packages**
  Demonstrates how libraries and packages are declared and used. {Lecture, Lab}
- **Interacting with Simulation**
  Describes how to interact with a simulation via text I/O. {Lecture}
- **Finite State Machine Overview**
  Provides an overview of finite state machines, one of the more commonly used circuits. {Lecture}
- **Mealy Finite State Machine**
  Describes how to implement a Mealy state machine in which the output is dependent on both the current state and the inputs. {Lecture}
- **Moore Finite State Machine**
  Demonstrates how to implement a Moore state machine in which the output is dependent on the current state only. {Lecture, Lab}
- **FSM Coding Guidelines**
  Describes the guidelines and recommendations for using one or more procedural blocks when coding a finite state machine. {Lecture}
- **Writing a Good Testbench**
  Explores how time-agnostic, self-checking testbenches can be written and applied. {Lecture, Lab}
- **Targeting AMD FPGAs and Adaptive SoCs**
  Focuses on implementation and chip-level optimization specific to AMD devices. {Lecture, Lab}