AMD
together we advance_

Designing with Verilog

LANG-VERILOG (v1.0)

Course Specification

Course Description

This course provides a thorough introduction to the Verilog 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

This course covers Verilog 1995 and 2001.

What's New for 2023.1

- All labs have been updated to the latest software versions

Level – FPGA 1

Course Details

- 3 days ILT/29

Course Part Number – LANG-VERILOG

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

Prerequisites

- Basic digital design knowledge

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:

- Write RTL Verilog code for synthesis
- Write Verilog test fixtures for simulation
- Create a finite state machine (FSM) by using Verilog
- Target and optimize AMD FPGAs and adaptive SoCs by using Verilog
- Use enhanced Verilog file I/O capabilities
- Run a timing simulation by using AMD Simprim libraries
- Create and manage designs within the Vivado Design Suite environment
- Download to the evaluation demo board

Course Outline

Day 1

- **Introduction to Verilog**
  Discusses the history of the Verilog language and provides an overview of the different features of Verilog. {Lecture}
- **Verilog Keywords and Identifiers**
  Discusses the data objects that are available in the Verilog language as well as keywords and identifiers. {Lecture}
- **Verilog Data Values and Number Representation**
  Covers what data values are in Verilog, as well as how to represent numbers in Verilog. {Lecture}
- **Verilog Data Types**
  Covers the various data types in Verilog. {Lecture}
- **Verilog Buses and Arrays**
  Covers buses and arrays in Verilog. {Lecture}
- **Verilog Modules and Ports**
  Describes both the syntax and hierarchy for a Verilog module, port declarations, and the difference between reg versus wire. {Lecture, Demo, Lab}
- **Verilog Operators**
  Shows the syntax for all Verilog operators. {Lecture}
- **Continuous Assignment**
  Introduces the Verilog continuous assignment statement. {Lecture}
- **Gate-Level Modeling**
  Introduces gate-level modeling in Verilog {Lecture}
- **Procedural Assignment**
  Provides an introduction to procedural assignments in Verilog, including their usage and restrictions. {Lecture}
- **Blocking and Non-Blocking Procedural Assignment**
  Introduces blocking and non-blocking assignment statements in Verilog. {Lecture, Lab}
- **Procedural Timing Control**
  Introduces the timing control methods that are used in procedural assignments. {Lecture}

Day 2

- **Verilog Control Structures: if-else**
  Describes the if/else control structure. {Lecture, Lab}
- **Verilog Control Structures: case**
  Describes the case control structure. {Lecture}
- **Verilog Loop Statements**
  Introduces the different types of Verilog loop statements. {Lecture}
- **Introduction to Verilog Testbenches**
  Introduces the concept of the Verilog testbench {Lecture, Lab}
- **System Tasks**
  Provides a basic understanding of system tasks. {Lecture}
- **Verilog Subprograms**
  Covers the use of subprograms in verification and RTL code to model functional blocks. {Lecture}
- **Verilog Functions**
  Describes functions, which are integral to reusable and maintainable code. {Lecture}
- **Verilog Tasks**
  Covers tasks in Verilog. {Lecture}
- **Verilog Compiler Directives**
  Describes Verilog compiler directives. {Lecture}
- **Verilog Parameters**
  Covers Verilog parameters and the local parameter concept. {Lecture, Lab}
- **Verilog Generate Statements**
  Introduces the Verilog generate statement. {Lecture}

Day 3

- **Timing Checks**
  Covers the timing check statements in Verilog and talks about the *specify* block. {Lecture}

- **Finite State Machines**
  Provides an overview of finite state machines, one of the more commonly used circuits. {Lecture}
- **Mealy Finite State Machine**
  Describes the Mealy FSM and how to code for it. {Lecture, Lab}
- **Moore Finite State Machine**
  Describes the Moore FSM and how to code for it. {Lecture, Lab}
- **FSM Coding Guidelines**
  Shows how to model an FSM of any complexity in Verilog and describes recommendations for performance and reliability. {Lecture}
- **Avoiding Race Conditions in Verilog**
  Describe what a race condition is and provides steps to avoid this condition. {Lecture}
- **File I/O: Introduction**
  Covers using basic and enhanced Verilog file I/O capabilities for more robust design verification. {Lecture}
- **File I/O: Read Functions**
  Covers Verilog file I/O read capabilities. {Lecture, Lab}
- **File I/O: Write Functions**
  Covers Verilog file I/O write capabilities. {Lecture}
- **Targeting AMD FPGAs and Adaptive SoCs**
  Focuses on implementation and chip-level optimization specific to AMD devices. {Lecture, Lab}
- **User-Defined Primitives**
  Describes user-defined primitives (UDPs). {Lecture}
- **Programming Language Interface**
  Introduces the programming language interface (PLI) in Verilog. {Lecture}