COURSE TABLE OF CONTENTS

HDL Code Verification and Debugging with MATLAB and Simulink

OEM: MathWorks • Code: MW-HDL-CODE-VERIFICATION-AND-DEBUGGING-W

COURSE DESCRIPTION

This two-day module introduces workflows for verifying and debugging HDL designs using MathWorks tools. It is designed for MATLAB and Simulink users interested in HDL verification only, verification and debugging on hardware, or those who also wish to generate HDL code. Testbench Generation Co-simulation FPGA-in-the-Loop FPGA Data Capture AXI Manager

AUDIENCE PROFILE

- IT professionals and technical staff seeking hands-on training in this course's subject area.

PREREQUISITES

- Fundamental knowledge of MATLAB and Simulink.

COURSE OBJECTIVES

By the end of this course, participants will be able to:

- Gain an overview of the verification and debugging workflows using MathWorks tools.
- Introduce advanced techniques for thorough HDL verification using model-based design, simulation, code coverage, and automated testbench generation.
- Verify and analyze HDL code by integrating MATLAB and Simulink into co-simulation workflows, enabling combined simulation of HDL and Simulink models.
- Prepare the necessary tools for verifying designs on an FPGA board. Use FPGA-in-the-Loop to validate implemented designs, whether they originate from generated or manually written HDL code.
- Capture live data from a running FPGA design to view and debug internal signals. Import the captured data into MATLAB or Simulink for comprehensive debugging and analysis.
- Access on-chip memory locations on an FPGA from MATLAB or Simulink using AXI Manager to perform read and write operations.

COURSE OUTLINE

Module 1: Day 1 of 2

Verification and Debugging Workflows for FPGA and ASIC Design

- Review the importance of a robust testbench.
- Explore workflows for verifying both generated and handwritten HDL code.
- Learn about hardware debugging and prototyping options.
- Install required Add-ons and Hardware Support Packages.

Testbench Generation

- Develop test stimuli based on the test plan, leveraging model coverage to ensure thoroughness.
- Perform verification of generated HDL code with an HDL simulator and a generated testbench.
- Use code coverage to identify untested portions of the code and improve test completeness.
- Verify generated HDL code in Simulink through co-simulation.
- Automatically generate a SystemVerilog DPI testbench from the full Simulink model and execute it for verification.

Co-Simulation

- Verify existing HDL code using MATLAB and Simulink through co-simulation.
- Integrate co-simulation models into simulation-based test environments with Simulink Test.
- Call MATLAB functions directly from an HDL simulator.
- Simulate HDL code alongside Simulink blocks using co-simulation blocks.

Module 2: Day 2 of 2

FPGA-in-the-Loop

- Identify appropriate use cases for FPGA-in-the-Loop (FIL) simulation.
- Set up hardware and software environments for FIL.
- Use HDL Workflow Advisor to perform FIL verification for auto-generated HDL code.
- Create a FIL block using the FIL Wizard and use it in MATLAB or Simulink.
- Accelerate FIL simulation time with frame-processing.
- Compare the design running on the board with a “golden reference model”.

FPGA Data Capture

- Integrate data capture capabilities into HDL IP and deploy it to FPGA hardware.
- Capture and analyze live data from FPGA boards using the FPGA Data Capture App.
- Configure trigger and capture conditions to optimize data acquisition.
- Automate the FPGA data capture workflow using MATLAB.
- Generate and configure FPGA Data Capture IP cores for existing HDL designs.
- Use the FPGA Data Reader block in Simulink to collect and visualize data from FPGAs.

Accessing AXI Registers on FPGA Using MATLAB and Simulink

- Access FPGA on-chip memory locations from MATLAB or Simulink using AXI Manager for reading and writing.
- Distinguish between AXI Manager and AXI Subordinate roles and their applications.
- Create and deploy an AXI Manager IP core within an FPGA design.
- Use the AXI Manager object in MATLAB to perform read and write operations on FPGA on-chip memory.