Generating HDL Code from Simulink

SciEngineer's training courses are designed to help organizations and individuals close skills gaps, keep up-to-date with the industry-accepted best practices and achieve the greatest value from MathWorks® and COMSOL® Products.

Generating HDL Code from Simulink

This two-day course shows how to generate and verify HDL code from a Simulink model using HDL Coder and HDL Verifier. This course focuses on preparing Simulink models for HDL code generation, fixed-point precision control, generating HDL code for multirate models, optimizing generated HDL code, interfacing external HDL code and verifying HDL code with cosimulation.

Prerequisites

Signal Processing with Simulink or equivalent experience using Simulink

DURATION	LEVEL
2 Days	Advanced
●●●	▬▬▬

TOPICS

Day 1

- Preparing Simulink Models for HDL Code Generation
- Fixed-Point Precision Control
- Generating HDL Code for Multirate Models

Day 2

- Optimizing Generated HDL Code
- Using Native Floating Point
- Interfacing External HDL Code with Generated HDL
- Verifying HDL Code with Cosimulation

Preparing Simulink Models for HDL Code Generation

OBJECTIVE: Prepare a Simulink model for HDL code generation. Generate HDL code and testbench for simple models requiring no optimization.

- Preparing Simulink models for HDL code generation
- Generating HDL code
- Generating a test bench
- Verifying generated HDL code with an HDL simulator

Fixed-Point Precision Control

OBJECTIVE: Establish correspondence between generated HDL code and specific Simulink blocks in the model. Use Fixed-Point Tool to finalize fixed point architecture of the model.

- Fixed-point scaling and inheritance
- Fixed-Point Designer workflow
- Fixed-Point Tool
- Command-line interface

Generating HDL Code for Multirate Models

OBJECTIVE: Generate HDL code for multirate designs.

- Preparing a multirate model for generating HDL code
- Generating HDL code with single or multiple clock pins
- Understanding and applying techniques used for clock domain crossing

Optimizing Generated HDL Code

OBJECTIVE: Use pipelines to meet design timing requirements. Use specific hardware implementations and share resources for area optimization.

- Generating HDL code with the HDL Workflow Advisor
- Meeting timing requirements via pipelining
- Choosing specific hardware implementations for compatible Simulink blocks
- Sharing FPGA/ASIC resources in subsystems
- Verifying that the optimized HDL code is bit-true cycle-accurate
- Mapping Simulink blocks to dedicated hardware resources on FPGA

Using Native Floating Point

OBJECTIVE: Implement floating point values and operations in your HDL code.

- Why and when to use native floating point
- Target-independent HDL code generation with HDL Coder
- Fixed-point vs. floating point comparison
- Optimization of floating point implementations

Interfacing External HDL Code with Generated HDL

OBJECTIVE: Incorporate hand-written HDL code and/or vendor party IP in your design.

- Interfacing external HDL code

Verifying HDL Code with Cosimulation

OBJECTIVE: Verify your HDL code using an HDL simulator in the Simulink model.

- Verifying HDL code generated with HDL Coder
- Comparing manually written HDL code with a "golden model"
- Incorporating HDL code into Simulink for simulation

sciengineer

Expand your knowledge