Generating HDL Code from Simulink

Training Objectives

This two-day course shows how to generate and verify HDL code from a Simulink® model using HDL Coder™ and HDL Verifier™.

Topics include:

- Preparing Simulink models for HDL code generation
- Generating HDL code and testbench for a compatible Simulink model
- Performing speed and area optimizations
- Integrating handwritten code and existing IP
- Verifying generated HDL code using testbench and cosimulation

Prerequisites

Signal Processing with Simulink or equivalent experience using Simulink

Products

- HDL Coder™
- HDL Verifier™
- Fixed-Point Designer™

Course Outline

Day 1 of 2

Preparing Simulink Models for HDL Code Generation (2.0 hrs)

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 (3.0 hrs)

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 (1.0 hrs)

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

Day 2 of 2

Optimizing Generated HDL Code (2.5 hrs)

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 (2.0 hrs)

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 (1.0 hrs)

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

- Interfacing external HDL code

Verifying HDL Code with Cosimulation (2.5 hrs)

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