Course Title:

Generating HDL Code from Simulink

Course Purpose

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

Pre- requisites

Signal Processing with Simulink course or equivalent experience using Simulink

- ✓ 2 training days
- ✓ Hours: 09:00-17:00
- ✓ Total training hours: 16

Teaching method

The course combines lectures, demonstrations and practical exercises in MATLAB, using original training books from MathWorks. The course is in Hebrew but the training materials are in English.

Training Center Systematics - Contact information:

Phone number: 03-7660111 Ext: 6 **Email:** training@systematics.co.il

Website: http://www.systematics.co.il/mathworks

סיסטמטיקס בע"מ
רחוב ראול ולנברג 4, תל אביב 69719
טלפון: 03-7660111
פקס: 03-7660120
מוקד שירות: 03-7660101
דוא"ל: info@systematics.co.il
www.systematics.co.il

Course Objective:

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: Use Fixed-Point Tool to convert your programmable logic design to fixed point.

- 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

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

Training Center Systematics - Contact information:

Phone number: 03-7660111 Ext: 6 **Email:** training@systematics.co.il

Website: http://www.systematics.co.il/mathworks

www.systematics.co.il
מוקד שירות: 03-7660101
דוא"ל: info@systematics.co.il
טלפון: 03-7660111
פקס: 03-7660120
סיסטמטיקס בע"מ
רחוב ראול ולנברג 4, תל אביב 69719

Using Native Floating Point

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

- Why and when to use native floating point
- Generating target-independent HDL code 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 an HDL component using Simulink

Training Center Systematics - Contact information:

Phone number: 03-7660111 Ext: 6 **Email:** training@systematics.co.il

Website: http://www.systematics.co.il/mathworks

www.systematics.co.il
מוקד שירות: 03-7660101
דוא"ל: info@systematics.co.il
טלפון: 03-7660111
פקס: 03-7660120
סיסטמטיקס בע"מ
רחוב ראול ולנברג 4, תל אביב 69719