Programming Xilinx Zynq SoCs with MATLAB and Simulink

Training Objectives

This hands-on, two-day course focuses on developing and configuring models in Simulink® and deploying on Xilinx® Zynq®-7000 All Programmable SoCs. The course is designed for Simulink users who intend to generate, validate, and deploy embedded code and HDL code for software/hardware codesign using Embedded Coder® and HDL Coder™.

A ZedBoard™ is provided to each attendee for use throughout the course. The board is programmed during the class and is yours to keep after the training.

Topics include:

- Zynq platform overview and environment setup
- Introduction to Embedded Coder and HDL Coder
- IP core generation and deployment
- Using AXI4 interface
- Processor-in-the-loop verification
- Data interface with real-time application
- Integrating device drivers
- Custom reference design

Prerequisites

Simulink for System and Algorithm Modeling (or *Simulink for Automotive System Design* or *Simulink for Aerospace System Design*). Knowledge of C and HDL programming languages.

Products

- Embedded Coder®
- HDL Coder™

Course Outline

Day 1 of 2

Zynq Platform Overview and Environment Setup (1.0 hrs)

Objective: Configure Zynq-7000 platform and MATLAB environment.

- Zynq-7000 overview
- Setting up Zynq platform and software
- Configuring MATLAB environment
- Testing connectivity to Zynq hardware

Introduction to Embedded Coder and HDL Coder (1.0 hrs)

Objective: Configure Simulink models for embedded code generation and effectively interpret the generated code.

- Architecture of an embedded application
- Generating ERT code
- Code modules
- Data structures in generated code

- Configuring a Simulink model for HDL code generation
- Using HDL Workflow Advisor

IP Core Generation and Deployment (2.0 hrs)

Objective: Use HDL Workflow Advisor to configure a Simulink model, generate and build both HDL and C code, and deploy to Zynq platform.

- Configuring a subsystem for programmable logic
- Configuring the target interface and peripherals
- Generating the IP core and integrating with SDK
- Building and deploying the FPGA bitstream
- Generating and deploying a software interface model
- Tuning parameters with External Mode

Using AXI4 Interface (2.0 hrs)

Objective: Use various AXI interfaces for data communication between processing system and programmable logic.

- AXI interface overview
- AXI4-Lite applications
- Using AXI4-Stream
- AXI4 performance considerations

Processor-in-the-Loop Verification (2.0 hrs)

Objective: Use processor-in-the-loop to verify the algorithm running on Zynq platform and profile the execution times in your production algorithm.

- Processor-in-the-loop (PIL) workflow on Zynq
- PIL verification with model reference
- Code execution profiling with PIL
- PIL considerations

Day 2 of 2

Data Interface with Real-Time Application (2.0 hrs)

Objective: Use the UDP interface to stream data between Simulink and the real-time application running on Zynq platform.

- Data interface overview
- Configuring UDP blocks for data streaming
- Synchronizing data between Simulink and Zynq
- Data interface with AXI Stream
- Design partitioning
- Data interface considerations

Integrating Device Drivers (2.0 hrs)

Objective: Develop device driver interfaces for integrating peripherals on processing system.

- Workflow for developing device drivers
- Using the Legacy Code Tool
- GPIO interface
- Cross-compiling device drivers

Custom Reference Design (2.0 hrs)

Objective: Create and package reusable IP for Vivado and register custom boards and reference designs.

- Motivations for a custom reference design
- Creating reusable IP for Vivado
- Reference design overview
- Customizing a reference design
- Registering board and custom reference design

Appendix A: Zynq Object and Linux Commands (0.5 hrs)

- Zynq object
- Linux commands

Appendix B: Fixed-Point Design (1.5 hrs)

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