sciengineer

Embedded Linux and System Integration for Zynq

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.

Embedded Linux and System Integration for Zynq

This two-day training course focuses on creating and customizing an embedded Linux system for custom target using Zynq. Topics discussed include creating a reference design in Vivado and SDK, software anatomy of a Zynq system, Zynq build system, building a custom Linux image for Zynq and integrating user space device drivers in Simulink.

Prerequisites

Programming Xilinx Zynq SoCs with MATLAB and Simulink.

DURATION	LEVEL
2 Days	Advanced
●●●	▬▬▬

TOPICS

Day 1

- Creating Reference Design in Vivado and SDK
- Software Anatomy of a Zynq System
- Zynq Build System

Day 2

- Zynq Build System (Continued)
- Integrating User Space Device Drivers in Simulink

Creating Reference Design in Vivado and SDK

OBJECTIVE: Create a Vivado® block diagram and SDK project to target PL and PS.

- Building a Vivado Block diagram to target PL
- Exporting hardware to SDK and creating board support package
- Creating a software application for ARM
- Automating build process through Tcl scripts

Software Anatomy of a Zynq System

OBJECTIVE: Understand various software components like FSBL, u-boot, kernel, userspace.

- Overview of embedded Linux
- Understanding various components of boot image

Zynq Build System

OBJECTIVE: Understand various software components to form a system image and an automation process.

- Understanding system boots
- Generating various binary files, including .elf, .bit, and open source build with buildroot
- Understanding storage device layout
- Using a MathWorks build system to have easy hooks for customization of bitstream/FSBL, devicetree and Kernel configuration

Zynq Build System (Continued)

OBJECTIVE: Create a custom linux image with device drivers for various PL and PS integrated peripherals.

- Updating devicetree to include new mappings
- Modifying kernel to turn on a driver
- Creating boot.bin and generating custom SD card image

Integrating User Space Device Drivers in Simulink

OBJECTIVE: Integrate device driver C code for peripherals in Simulink to communicate with the custom Linux image.

- Overview of the generated C code main function, scheduler timing and POSIX threads
- Creating a custom System object™
- Using coder.ceval and System objects for C code integration
- Interacting with custom Linux images from Simulink
- Creating a standalone application as part of boot image

sciengineer

Expand your knowledge