AMD
together we advance_

Embedded Systems Software Design

EMBD-SW (v1.0)

Course Specification

Course Description

This course introduces the concepts, tools, and techniques required for software design and development for the AMD Zynq™ System on a Chip (SoC), Zynq UltraScale+™ MPSoC, and Versal™ adaptive SoC architectures using the AMD Vitis™ Unified IDE.

The focus is on:

- Reviewing the basics of Vitis Unified IDE
- Customizing board support packages (BSPs) for resource access and management of the AMD Standalone library
- Utilizing device drivers effectively
- Developing software applications for the available processors
- Debugging and integrating user applications
- Employing best practices to enable good design decisions

What's New for 2024.1

- All labs have been updated to the latest software versions

Level – Embedded Software 3

Course Details

- 3 days ILT/29

Course Part Number – EMBD-SW

Who Should Attend? Software design engineers interested in system design and implementation and software application development and debugging using the AMD Standalone library

Prerequisites

- C or C++ programming experience, including general debugging techniques
- Conceptual understanding of embedded processing systems, including device drivers, interrupt routines, writing and modifying scripts, user applications, and boot loader operation

Software Tools

- Vitis Unified IDE 2024.1
- Vivado™ Design Suite 2024.1

Hardware

- Architectures: Zynq 7000 SoC (Cortex-A9 processor) and Zynq UltraScale+ MPSoC (Cortex-A53 and Cortex-R5 processor)
- Demo board: Zynq UltraScale+ MPSoC ZCU104 or Versal adaptive SoC VCK190 board

* This course focuses on the Zynq 7000 SoC and the Zynq UltraScale+ MPSoC architectures. Check with your local Authorized Training Provider for the specifics of the in-class lab board or other customizations.

After completing this comprehensive training, you will have the necessary skills to:

- Implement an effective software design environment for an AMD embedded system using the AMD software development tools
- Write a basic bare-metal or Linux user application using the AMD Vitis Unified IDE and run it on an embedded system
- Use AMD debugger tools to troubleshoot user applications
- Apply software techniques to improve operability
- Maintain and update software projects with changing hardware

Course Outline

Day 1

- **Overview of Embedded Software Development**
  Overview of the process for building a user application. {Lecture}
- **Embedded UltraFast Design Methodology**
  Outlines the different elements that comprise the Embedded Design Methodology. {Lecture, Demo}
- **Zynq 7000 SoC Architecture Overview**
  Overview of the Zynq 7000 SoC architecture. {Lecture, Demo, Lab}
- **Zynq UltraScale+ MPSoC Architecture Overview**
  Overview of the Zynq UltraScale+ MPSoC architecture. {Lecture, Demo, Lab}
- **Zynq UltraScale+ MPSoC Software Environments**
  Describes the software development environments for Zynq UltraScale+ MPSoCs. {Lecture}
- **Driving the AMD Vitis Unified IDE**
  Introduces the terminology and features of the Vitis Unified IDE and talks about the basic behaviors required to drive the Vitis Unified IDE to generate a C/C++ application. {Lecture, Lab}
- **Debugging Using the AMD Vitis Unified IDE**
  Describes the basics of actually running the Vitis Unified IDE system debugger and illustrates the debugging process. {Lecture, Lab}

Day 2

- **Bare-metal Application Development**
  Covers the various software components, or layers, supplied by AMD that aid in the creation of low-level software. The basic bare-metal application development flow is also discussed. {Lecture, Demo, Lab}
- **FAT File System for Standalone**
  Introduces the FAT file system (FFS) from the Standalone/Bare-metal library. The FFS provides drivers and utilities for effectively converting a region of memory into a file system. {Lecture, Lab}
- **Using Linker Scripts**
  Overview of the purpose and typical use of a linker script. {Lecture}
- **Introduction to Interrupts**
  Introduces the concept of interrupts, basic terminology, and generic implementation. {Lecture}
- **Software Interrupts: Writing**
  Describes many of the considerations that a software coder must take into account when supporting interrupts. {Lecture, Lab}

Day 3

- **Operating Systems: Introduction and Concepts**
  Introduces the concept of the operating system and provides a simplified view into the generic way that operating systems work. {Lecture}
- **Linux: A High-Level Introduction**
  Introduces the Linux operating system, a brief history, and how to use it. {Lecture}
- **Linux Software Application Development**
  Highlights important parts of the underlying Linux system as it pertains to applications. {Lecture, Demo, Lab}

- **Driving the PetaLinux Tool**
  Introduces the basic concepts required to build an application using the PetaLinux tool. {Lab}
- **Booting Overview**
  Describes the main points to how booting a processor is handled for Zynq SoC devices and MicroBlaze processors. {Lecture, Lab}
- **Software Profiling Overview**
  Introduces the purpose and techniques for profiling a user application. {Lecture, Demo, Lab}
- **Understanding Device Drivers**
  Explains the concept of a device driver and how it is used by embedded systems. {Lecture}
- **Custom Device Drivers**
  Describes how to successfully write a custom device driver. {Lecture}
- **Debugging Using Cross-Triggering**
  Illustrates how hardware-software cross-triggering techniques can uncover issues. {Lecture, Lab}