AMD XILINX

Designing with Versal AI Engines

Course Description

This course is a combination of two courses related to the Versal™ AI Engine which are often taken back to back. This combination offers students the savings of time and money by removing the redundancies between the two courses to focus on unique content and a streamlined training experience:

Part-1: Designing with Versal AI Engine 1: This module describes the Versal™ AI Engine architecture, how to program the AI Engines (single kernel programming and multiple kernel programming using data flow graphs), the data communications between the PL and AI Engines, and how to utilize the AI Engine library for faster development.

The emphasis of this module is on:

- Illustrating the AI Engine architecture
- Designing single AI Engine kernels
- Designing multiple AI kernels using data flow graphs with the Vitis™ IDE
- Reviewing the data movement between AI Engines, between AI Engines via memory and DMA, and between AI Engines to programmable logic (PL)
- Analyzing and debugging kernel performance.

Part-2: Designing with Versal AI Engine 2: This module describes the system design flow and interfaces that can be used for data movements in the Versal™ AI Engine. It also demonstrates how to utilize the advanced MAC intrinsics, AI Engine library for faster development and advanced features in static data flow graph implementation such as using streams, cascade stream, buffer location constraints, run-time parameterization and APIs to update and/read run-time parameters.

The emphasis of this course is on:

- Implementing a system-level design flow (PS + PL + AIE) and the supported simulation
- Using an interface for data movement between the PL and AI Engine
- Utilizing advanced MAC intrinsics to implement filters
- Utilizing the AI Engine library for faster development
- Applying advanced features for optimizing a system-level design

Course Details

- 30 hours – 5 days (6 hours each day)

Course Part Number – ACAP-VIE1/2

Who Should Attend? – Software and hardware developers, system architects, and anyone who needs to accelerate their software applications using Xilinx devices.

Prerequisites

- Comfort with the C/C++ programming language
- Software development flow
- Experience using Vitis software for application acceleration development flow

Software Tools

- Vitis unified software platform 202**0.2**

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

Describe the Versal ACAP architecture at a high level

- Describe the various engines in the Versal ACAP device and the motivation behind the AI Engine
- Describe the architecture of the AI Engine
- Describe the memory access structure for the AI Engine
- Describe the full application acceleration flow with the Vitis tool
- Enumerate the toolchain for Versal AI Engine programming
- Explain what intrinsic functions are
- Program a single AI Engine kernel
- Program multiple AI Engine kernels using static data flow (SDF) graphs
- Describe the system-level flow, which includes PS + PL + AIE (SW-HW-SW) designs
- Describe the supported emulation for a system-level design
- Describe the data movement between the PS, PL, and AI Engines
- Describe the implementation of the AI Engine core and programmable logic
- Implement a system-level design for Versal ACAPs with the Vitis tool flow
- Utilize advanced MAC intrinsic syntax and application-specific intrinsics such as DDS and FFT
- Utilize the AI Engine DSP library for faster development
- Apply location constraints on kernels and buffers in the AI Engine array
- Apply runtime parameters to modify application behavior
- Debug a system-level design

Course Outline

Session 1

- **Overview of Versal ACAP Architecture**
  Provides an overview of the Versal architecture at a high level and describes the various engines in the Versal ACAP, such as the Scalar Engines, Adaptable Engines, and Intelligent Engines. Also describes how the AI Engine in the Versal ACAP meets many dynamic market needs. {Lecture}
- **Introduction to the Versal AI Engine Architecture**
  Introduces the architecture of the AI Engine and describes the AI Engine interfaces that are available, including the memory, lock, core debug, cascaded stream, and AXI-Stream interfaces. {Lecture}
- **Versal AI Engine Memory and Data Movement**
  Describes the memory module architecture for the AI Engine and how memory can be accessed by the AI Engines in the AI Engine arrays. {Lecture}
- **Versal AI Engine Tool Flow**
  Reviews the Vitis tool flow for the AI Engine and demonstrates the full application acceleration flow for the Vitis platform. {Lecture, Lab}

Session 2

- **Application Partitioning on Versal ACAPs 1**
  Covers what application partitioning is and how an application can be accelerated by using various compute engines in the Versal ACAP. Also describes how different models of computation (sequential, concurrent, and functional) can be mapped to the Versal ACAP. {Lecture}
- **Application Partitioning on Versal ACAPs 2**
  Explains how image and video processing can be targeted for the Versal ACAP by utilizing the different engines.
- **Data Types: Scalar and Vector Data Types**
  Provides an AI Engine functional overview and identifies the supported vector data types and high-

width registers for allowing single-instruction multiple-data (SIMD) instructions {Lecture}

- **Intrinsic Functions**
  Describes what intrinsic functions are, the three types of vector management operations using intrinsic functions (load and store, element conversion, and lane insertion/extraction), multiplication functions, and application-specific functions. {Lecture}

Session 3

- **Window and Streaming Data APIs**
  Describes window and streaming APIs and reviews the various window operations for kernels. Also discusses using overlapping data and various data movement use cases. {Lecture}
- **The Programming Model: Single Kernel**
  Reviews the AI Engine kernel programming flow for programming and building a single kernel. Also illustrates the steps to create, compile, simulate, and debug a single kernel program. {Lecture, Lab}
- **The Programming Model: Introduction to the Data Flow Graph**
  Provides the basics of the data flow graph model and graph input specifications for AI Engine programming. Also reviews graph input specifications, such as the number of platforms and ports. {Lecture}
- **The Programming Model: Multiple Kernels Using Graphs**
  Describes the static data flow (SDF) graph and demonstrates the steps to create a graph and set the runtime ratio and graph control APIs from the main application program. {Lecture, Lab}

Session 4

- **ACAP Data Communications 1**
  Describes the implementation of AI Engine cores and the programmable logic (PL). Implement the functions in AI Engine that take advantage of low power. {Lecture}
- **ACAP Data Communication 2**
  Describes the programming model for the implementation of stream interfaces for the AI Engine kernels and PL kernels. Lists the stream data types that are supported by AI Engine and PL kernels. {Lecture}
- **System Design Flow**
  The Vitis compiler flow lets you integrate your compiled AI Engine design graph (libsdf.a) with additional kernels implemented in the PL region of the device, including HLS and RTL kernels, and link them for use on a target platform. You can call these compiled hardware functions from a host program running in the Arm® processor in the Versal device or on an external x86 processor. {Lecture, Lab}
- **Introduction to Advanced Intrinsic Functions**
  Describes how to implement filters using advanced intrinsics functions for various filters, such as non-symmetric FIR, symmetric FIR, half-band decimators. {Lecture}

Session 5

- **Versal AI Engine DSP Library Overview**
  Provides an overview of the available DSP library which enables faster development and comes with ready-to-use example designs which helps with using the library and tools. {Lecture, Labs}
- **Advanced Graph Input Specifications 1**
  Learn advanced features such as using initialization functions, writing directly using streams from the AI Engine, cascade stream, core location constraints, and buffer location constraints. {Lecture}
- **Advanced Graph Input Specifications 2**
  Describes how to implement runtime parameterization, which can be used as adaptive feedback and switching functionality dynamically. {Lecture, Lab}
- **Versal AI Engine Application Debug and Trace** Shows to how to debug the AI Engine application running on the Linux OS and how to debug via hardware emulation that allows simulation of the application. {Lecture}