AMD
together we advance_

Designing with Versal AI Engine: Graph Programming with AI Engine Kernels - 2

AIE-GRAPH (v1.0)

Course Specification

Course Description

This course describes the system design flow and interfaces that can be used for data movement in the Versal™ AI Engine. It demonstrates how to utilize AI Engine APIs and the AI Engine DSP library for faster development.

In addition, advanced features in adaptive data flow (ADF) graph implementation, such as using streams, cascade streams, buffer location constraints, runtime parameterization, and APIs to update and read runtime parameters, are covered. The course also highlights how to utilize the Vitis™ Model Composer tool for AI Engine designs.

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 AI Engine APIs for arithmetic operations and advanced MAC intrinsics to implement filters
- Utilizing the AI Engine DSP library for faster development
- Applying runtime parameters for an AI Engine design
- Utilizing the Vitis Model Composer tool for AI Engine designs

What's New for 2024.1

- Versal Adaptive SoC - Application Mapping and Partitioning module: Updated with the system design planning flow
- All labs have been updated to the latest software versions

Level – ACAP 3

Course Details

- 2 days ILT/19

Course Part Number – AIE-GRAPH

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

Prerequisites

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

Software Tools

- Vitis Unified IDE 2024.1
- Vitis Model Composer 2024.1

Hardware

- Architecture: Versal adaptive SoCs

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

- Describe system-level application partitioning for AMD Versal devices
- Describe the data movement between the PS, PL, and AI Engines as well as the clocking specifications
- Describe and implement a system-level design integrating PS + PL + AI Engines for Versal adaptive SoCs using supported emulation and the Vitis tool
- Describe the Vitis export to Vivado™ flow as well as the AI Engine reset and reload methodology
- Utilize the AI Engine DSP library for faster development
- Apply location constraints on kernels and buffers in the AI Engine array
- Apply runtime parameters in a graph to modify application behavior
- Debug and analyze an AI Engine system design using Vitis debug and trace
- Utilize the AI Engine library in Vitis Model Composer for AI Engine development

Course Outline

Day 1

Design Analysis

- **Versal Adaptive SoC: Application Mapping and Partitioning (Review)**

  Covers what application partitioning is and how an application can be accelerated by using various compute engines in the Versal device. Also describes how different models of computation (sequential, concurrent, and functional) can be mapped to the Versal adaptive SoC. {Lecture}
- **Versal Adaptive SoC: Application Partitioning and Implementation**

  Explains how an application like image and video processing can be targeted for the Versal device by utilizing the different compute elements (processor system, programmable logic, and AI Engines). Also describes the AI Engine development flow. {Lecture}

Versal AI Engine Data Movement

- **Versal Adaptive SoC: Data Communications 1**

  Describes the implementation of AI Engine and programmable logic (PL) kernels and how to implement the functions in the AI Engine that take advantage of low power. {Lecture}
- **Versal Adaptive SoC: Data Communications 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, Lab}

Vitis Tool Flow

- **System Design Flow**

  Demonstrates the Vitis compiler flow to integrate a compiled AI Engine design graph (libadf.a) with additional kernels implemented in the PL region of the device (including HLS kernels) and link them for use on a target platform. These compiled hardware functions can then be called from a host program running in the Arm® processor in the Versal device or on an external x86 processor. {Lecture, Lab}
- **System Design Flow: Models and Methodology**

  Reviews the system simulation models and illustrates the Vitis export to Vivado flow. Also describes the AI Engine reset and reload methodology. {Lecture, Lab}

The Programming Model

- **Introduction to AI Engine APIs for Arithmetic Operations**

  Describes the Versal AI Engine APIs for arithmetic, comparison, and reduction operations. Also describes the matrix multiplication, FFT, and special multiplication operations. For advanced users, covers how to implement filters using advanced intrinsics functions for various filters, such as non-symmetric FIRs, symmetric FIRs, or half-band decimators. {Lecture}

Day 2

Libraries

- **Introduction to the AI Engine DSP Library**
  Provides an overview of the available DSP library, which enables faster development and comes with ready-to-use example designs that help with using the library and tools. {Lecture, Labs}

The Programming Model

- **Advanced Graph Input Specifications**
  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 and Runtime Parameters**
  Describes how to implement runtime parameterization, which can be used as adaptive feedback and to switch functionality dynamically. {Lecture, Lab}

Debugging

- **Versal AI Engine Application Debug and Event Trace**
  Shows how to debug the AI Engine design in an application and how to debug via hardware emulation. {Lecture}

Vitis Model Composer

- **Vitis Model Composer for AI Engine Design**
  Describes the Vitis Model Composer tool and how to use the libraries available with the tool for AI Engine design development. {Lecture, Lab}