Designing with Versal AI Engine 3: Kernel Programming and Optimization

ACAP 4

ACAP-AIE3 (v1.0)

Course Specification

Course Description

This course covers the advanced features of the Versal™ ACAP AI Engine, including debugging an application in the Vitis™ unified software platform, using filter intrinsics, implementing a system design in hardware, and optimizing an AI Engine kernel program.

The emphasis of this course is on:

- Reviewing the advanced features of the Versal ACAP AI Engine architecture
- Optimizing AI Engine kernels using compiler directives, programming style, and efficient movement of data
- Describing C++ kernel template functionality
- Identifying the different types of kernel instance states
- Using AI Engine filter intrinsics and programming a FIR filter using filter intrinsics
- Debugging applications using the Vitis unified software platform
- Describing the Vitis Model Composer tool for AI Engine kernel development

What's New for 2021.1

- System Generator for DSP is now part of Vitis Model Composer
- All labs have been updated to the latest software versions

Level – ACAP 4

Course Details

- 2 days ILT or 16 hours OnDemand
- 13 lectures
- 4 labs

Course Part Number – ACAP-AIE3

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
- Vitis software for application acceleration development flow
- *Designing with Versal AI Engine 1: Architecture and Design Flow*
- *Designing with Versal AI Engine 2: Graph Programming with AI Engine Kernels*

Software Tools

- Vitis unified software platform 2021.1

Hardware

- Architecture: Xilinx Versal ACAPs

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

- Utilize various AI Engine kernel optimization techniques, such as compiler directives, software pipelining, coding for performance, and core utilization
- Apply C coding guidelines for performance improvement, including function inlining, pointer restricting, and code shuffling
- Identify and debug the various problems that arise in application development
- Implement an AI Engine kernel using intrinsics for a symmetric FIR with mul4_sym and mac4_sym
- Implement an AI Engine kernel using a non-symmetric FIR with mul4_nc and mac4_nc
- Debug an application using the simulation debugging methodology and event traces
- Use Vitis Model Composer for AI Engine kernel development and modeling of a heterogeneous device

Course Outline

Day 1

- **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 Data Movement and Interfaces**

  Describes the memory module architecture for the AI Engine and how memory can be accessed by the AI Engines in the AI Engine arrays. Also reviews the AI Engine array interfaces. {Lecture}
- **Overview of AI Engine Kernel Optimization**

  Explains the various AI Engine kernel optimization techniques, such as compiler directives, software pipelining, coding for performance, and core utilization. {Lecture}
- **AI Engine Kernel Optimization – Compiler Directives**

  Describes the usage of compiler directives for loop unrolling, loop flattening, and software pipelining to help improve the performance of AI Engine kernels. {Lecture}
- **AI Engine Kernel Optimization – Coding Style**

  Covers the C coding guidelines for performance improvement, including function inlining, pointer restricting, and code shuffling. Also covers calculating AI Engine utilization for the kernels to help improve performance. The lab illustrates applying kernel optimization techniques such as the restrict keyword, custom pragmas, and code restructuring. {Lecture, Lab}
- **Advanced C++ Kernel Programming**

  Provides an overview of C++ kernel template functionality and the different types of states and kernel instance states using C++ classes. Also covered are kernel instance states with scalar parameters in a constructor as well as kernel instance states with array parameters in a constructor. {Lecture, Lab}

Day 2

- **Vector Data Types (Review)**

  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}
- **AI Engine Symmetric Filter Implementation**

  Describes advanced MAC intrinsic syntax, including the intrinsics for symmetric FIR implementation, such as mul4_sym and mac4_sym. Also provides guidelines for choosing the right fixed-point intrinsics for a FIR filter. {Lecture, Lab}
- **AI Engine Non-Symmetric Filter Implementation**

  Describes the intrinsics for non-symmetric FIR implementations, such as mul4_nc and mac4_nc. Also provides guidelines for choosing the right intrinsics for a FIR filter. {Lecture}
- **Floating-Point Operations**

  Reviews the floating-point operations fpmul, fpmac, and fpmsc as well as the fully configurable, floating-point intrinsics fpmac_conf. {Lecture}
- **Debugging AI Engine Applications 1**

  Describes the application simulation debugging methodology as well as debugging with event traces, such as AI Engine events,

DMA events, lock events, and stream events. Also demonstrates how to visualize these events in the Vitis unified software platform. {Lecture, Lab}

- **Debugging AI Engine Applications 2 (Use Cases)**
  Reviews various use cases of problems that arise, such as memory conflicts and deadlock analysis. Also covers performance analysis (profiling) in hardware. {Lecture}
- **Vitis Model Composer for AI Engine Development**
  Introduces Vitis Model Composer and how it can help with developing AI Engine kernels and modeling a heterogeneous device. {Lecture}

Register Today

Visit the Xilinx Customer Training Center to view schedules and register online