Designing with Versal AI Engine: Kernel Programming and Optimization (3)

AIE-KERNEL (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
- Programming a FIR filter using AI Engine APIs
- Debugging applications using the Vitis unified software platform

What's New for 2022.1

- Modules and labs have been updated based on the enhanced programming model
- All labs have been updated to the latest software versions

Level – ACAP 4

Course Details

- 2 days

Course Part Number – AIE-KERNEL

Who Should Attend? – Software and hardware developers, system architects, and anyone who needs to accelerate their software applications using AMD-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: Architecture and Design Flow (1)*
- *Designing with Versal AI Engine: Graph Programming with AI Engine Kernels (2)*

Software Tools

- Vitis unified software platform 2022.1

Hardware

- Architecture: Versal ACAPs

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

- Utilize various Versal 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 AI Engine APIs for a symmetric FIR with aie::sliding_mul_sym_xy_ops
- Implement an AI Engine kernel using a non-symmetric FIR with aie::sliding_mul_sym_xy_ops
- Debug an application using the simulation debugging methodology and event traces

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 and Asymmetric Filter Implementation**
  Describes AI Engine APIs for symmetric and asymmetric FIR implementation, such as aie::sliding_mul_sym_xy_ops operators. Also, provides an overview of the DSP library, which can help with creating filters more easily and faster. {Lecture, Lab}
- **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}
- **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, Lab}
- **Versal AI Engine DSP Library Overview [Optional]**
  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}

© 2022 Xilinx, Inc. All rights reserved. All Xilinx trademarks, registered trademarks, patents, and disclaimers are as listed at http://www.xilinx.com/legal.htm.
All other trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

Appendix:

- **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}
- **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}
- **Designing and Vectorizing an AI Engine with System View Visual System Integrator**
  Introduces Visual System Integrator, a customized tool from System View used for AI Engine design, code vectorization, and ADF graph generation.

© 2022 Xilinx, Inc. All rights reserved. All Xilinx trademarks, registered trademarks, patents, and disclaimers are as listed at http://www.xilinx.com/legal.htm.
All other trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.