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 AMD Versal™ adaptive SoC AI Engine, including kernel function development, optimizing an AI Engine kernel program, using AI Engine APIs and filter intrinsics, and debugging an application in the Vitis™ Unified IDE.

The emphasis of this course is on:

- Reviewing the features of the Versal device 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 FIR filters using AI Engine APIs
- Debugging applications using the Vitis Unified IDE

What's New for 2024.1

- AI Engine Symmetric and Asymmetric Filter Implementation module: Updated the special multiplication functions that are supported for AIE-ML
- All labs have been updated to the latest software versions

Level – ACAP 4

Course Details

- 2 days live instructor led training (online or in person)
- 11 lectures
- 6 labs

Price – $1,600 or 16 AMD Training Credits

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 devices

Prerequisites

- Comfort with the C/C++ programming language
- Vitis software for application acceleration development flow
- Designing with Versal AI Engine: Quick Start
- Designing with Versal AI Engine: Architecture and Design Flow - 1
- Designing with the Versal Adaptive SoC: Design Methodology
- Designing with Versal AI Engine: Graph Programming with AI Engine Kernels - 2

Software Tools

- Vitis Unified IDE 2024.1

Hardware

- Architecture: Versal adaptive SoCs

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 implement the different types of kernel instance states using C++ kernel development
- Implement AI Engine kernels using AI Engine APIs for symmetric and non-symmetric FIRs
- Debug an application using the simulation debugging methodology and event traces

Course Outline

Day 1

- **AI Engine and Memory Module Architecture**
  Introduces the architecture of the AI Engine and describes the memory module architecture for the AI Engine. {Lecture}
- **Versal AI Engine Data Movement and Interfaces**
  Describes the data movement and memory access by the AI Engines in the AI Engine arrays. Also reviews the AI Engine interfaces that are available, including the lock, core debug, cascaded stream, and AXI-Stream 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 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 – Event Trace**
  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 – 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}
- **Introduction to the AI Engine DSP Library [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}

© Copyright 2024 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo, Versal, Vitis, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

Appendix:

- **AI Engine Symmetric Filter Implementation Using Intrinsics**
  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 Using Intrinsics**
  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 Using Intrinsics**
  Reviews the floating-point operations fpmul, fpmac, and fpmsc as well as the fully configurable, floating-point intrinsics fpmac_conf. {Lecture}

Register Today

Morgan Advanced Programmable Systems, Inc. (Morgan A.P.S.) delivers public and private courses in locations throughout the central US region; including Iowa, Illinois, Kansas, Minnesota, Missouri, Nebraska, North Dakota, South Dakota, and Wisconsin.

Visit **morgan-aps.com/training**, for full course schedule and training information.

Morgan Advanced Programmable Systems

- You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and Xilinx training credits.

Student Cancellation Policy

- Student cancellations received more than 7 days before the first day of class are entitled to a 100% refund. Refunds will be processed within 14 days.
- Student cancellations received less than 7 days before the first day of class are entitled to a 100% credit toward a future class.
- Student cancellations must be sent here.

Morgan A.P.S. Course Cancellation Policy

- We regret from time-to-time classes will need to be rescheduled or cancelled.
- In the event of cancellation, live on-line training may be offered as a substitute.
- Morgan A.P.S. may cancel a class up to 7 days before the scheduled start date of the class; all students will be entitled to a 100% refund.
- Under no circumstances is Morgan A.P.S. responsible or liable for travel, lodging or other incidental costs. Please be aware of this cancellation policy when making your arrangements.
- For additional information or to schedule a private class contact us here.

Online or in person training with real hardware

- Morgan Advanced Programmable Systems, Inc. has set up a training VPN where engineer participants can take classes online using the same computers and devCards used during in-person training.
- Even better, and upon request, you can use these computers after hours on training days to experiment with labs. This is not possible for in-person training.
- Additionally, just like in-person training, the laptops and devCards, tools, OS, and licensing are set up in advance.
- In some ways, live online-training is better than in-person…for example, you can grant the instructor permission to look at your Vivado, PetaLinux terminal, or Vitis for extended periods of time if your lab is not going exactly has planned to a missed step.
- This is often more comfortable than two engineers crowding around a laptop screen.
- Taking remote training also allows you to learn some tips and tricks for working remote. Whether your devCard is in the lab down the hall, or across the world via VPN, you can control your Xilinx based device quickly and efficiently.

© Copyright 2024 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo, Versal, Vitis, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.