www.kaztek.com

WS_BFDEV

System Development and Programming with the ADI Blackfin Processor Family

Course Name:	System Development and Programming with the ADI Blackfin Processor Family
Course Code:	WS_BFDEV
Course Description:	This is a practical and interactive course that is designed to systematically teach how to use the Blackfin processor to its fullest potential. Emphasis is placed on understanding the steps required to create an efficient Blackfin CPU based system in the way that ADI had intended the processor to be used. Several hands on exercises provide an opportunity for the instructor to work one on one with the attendee. Throughout the workshop, attendees are encouraged to ask questions. The VisualDSP++ IDDE is covered in detail, including topics on projects and project configuration, the build process, and debug features. Tools based optimizations including compiler and linker optimization are covered. The salient features of System Services and Device Drivers are first introduced. Later sections on specific peripherals or system features will connect back to the related Services or Driver API. An understanding of the Blackfin architecture will enable getting the best performance out of the processor. Architecture topics covered include loop/branch optimization and interrupt handling, L1 Memory configuration (ie L1 SRAM and Cache), specialized instructions including the quad 8-bit Video ALU operations, and DMA operation between peripherals and memory, as well as from memory to memory. The usage and capabilities of the various I/O and timer peripherals are discussed in detail. A section on booting covers what happens during the boot process, creating boot image files, and discussing how to get them into the target system. Hardware considerations such as operation of the power management and reset are also discussed. Hardware development tools, such as evaluation boards and ICE's are also covered, including setting up hardware debug sessions and other hardware debug topics. An introduction into VDK (Visual DSP Kernel) is also covered in the workshop. Throughout the course, a number of hands on exercises will take the attendee through the various aspects of the software development process. Topics covered through exercises include setting up and building projects, C language programming (eg using intrinsics, exploring different data types), various optimizations (eg compiler, linker, mixed C/assembly), code debugging (eg profiling, plotting, pipeline analysis), simulation (eg DMA, interrupts, stream I/O), and 'C' programming support (eg System Services and Device Drivers). One set of exercises will focus on bringing the various elements together (eg peripheral setup, DMA, interrupts, etc) in the setup of a typical application framework, initially in straight C and later using the System Services and Device Drives programming model.
Goals and Objectives:	The main course objective is to instill to the attendee a high level of confidence to create an efficient Blackfin® Processor based system. By being exposed to the various capabilities and features of the Blackfin processor and VDSP tool chain, the attendee will be better armed to tackle any issues that arise in their specific system implementations.
Pre-requisites:	Previous embedded microprocessor experience would be an asset (hardware and/or software). It is also highly recommended to take in the Visual Learning & Development (VLD) Video Tutorial entitled "Blackfin® Core Architecture"
Target Audience:	System Designers needing to make informed decisions on design tradeoffs, Hardware Designers needing to develop external interfaces and low level code, and Code Developers needing to know how to get the highest performance from their algorithms
Duration:	3.5 days

3 Data Formats

3.1 Supported Data Types

3.2 Fixed Point Numbering Formats

3.2.1 Built-in Support for Fractional Data Types

3.3 Floating Point

3.4 Exercise – Floating Point Operations

4 Blackfin Core Review

4.1 Registers

4.1.1 Core Registers

4.1.2 Memory Mapped Registers

4.1.3 Register File

4.2 Arithmetic Units

4.2.1 Features

4.2.2 Arithmetic Logic Unit (ALU)

4.2.3 Multiplier/ Accumulator (MAC)

4.2.4 Shifter

4.2.5 Example Instructions

4.3 Fetching Data

4.3.1 Data Address Generators (DAG) Features

4.3.2 Data Move Examples

4.3.3 Address Pointer Manipulation

4.3.4 Stack Operations

4.3.5 DSP Addressing

4.3.5.1 Circular Buffering

4.3.5.2 Bit Reversal

4.4 8-bit ALU Instructions

4.4.1.1 Compiler builtin support for Video ALU

4.5 Vector Operations

4.6 Issuing Parallel Instructions

4.7 Other Code Optimization Topics

4.8 Instructor Led Assembly Code Exercise

18 Optimization Topics

18.1 Overview
18.2 General Approach to Optimization
18.3 Algorithmic considerations
18.4 Getting to know the Compiler
18.4.1 Optimization Switches and pragma’s
18.4.2 Built-in functions
18.4.3 Example compiler output
18.4.4 PGO (Profile Guided Optimization)
18.4.5 C++ considerations
18.4.6 Summary
18.5 Where to start optimizing?
18.5.1 Using the profiler effectively
18.6 Assembly Language Interfacing
18.7 Video Frameworks
18.7.1 Challenges
18.7.2 Common Frameworks

19 Hardware Tools

19.1 EZKITs
19.1.1 Overview with Part Numbers
19.1.2 EZKIT Extender Boards
19.1.3 EZKIT Debug Sessions
19.1.4 Application examples
19.2 In Circuit Emulators (ICE)
19.2.1 Configuration of a Debug Target
19.2.2 Emulator Debug Sessions

20 VDK Introduction