



RA9 - CORTEX-A73 implementation, ARM Architecture V8

This course covers the Cortex-A73 and AARCH64

Objectifs

- This course aims to highlight the new features offered by the V8 architecture.
- It has been developed for engineers developing low level software and engineers in charge of IP hardening.
- First, an overview of Cortex-A73 is provided, to highlight the differences between a Cortex-A15/Cortex-A7 hardware platform based on CCI-400 and a Cortex-A73/Cortex-A53 hardware platform based on CCN-504.
- The new exception mechanism is described.
- The enhancements regarding the LPAE are detailed.
- New A64 assembler instructions are explained through practical examples.
- The AAPCS64 is also covered.
- The course also details the new debug ARM V8 features.
- Cortex-A73 hardware implementation is explained, particularly the low power states.

Prerequisites and related courses

- Knowledge of a 32-bit ARM CPU is recommended
- See also [RI0 - AXI3 / AXI4 INTERCONNECT](#) course and [RC1 - NEON-v7 programming](#) course that describes the advanced SIMD (NEON) architecture and instruction set

Course Environment

- Theoretical course
 - PDF course material (in English) supplemented by a printed version for face-to-face courses.
 - Online courses are dispensed using the Teams video-conferencing system.
 - The trainer answers trainees' questions during the training and provide technical and pedagogical assistance.
- At the start of each session the trainer will interact with the trainees to ensure the course fits their expectations and correct if needed

Target Audience

- Any embedded systems engineer or technician with the above prerequisites.

Course Outline

First Day

Overview of Cortex-A73

- Cortex-A73 architecture
- Intra-cluster cache coherency
- Inter-cluster cache coherency through AXI4 ACE
- ACP port
- Programmer's model

Introduction to ARM Architecture V8

- Enhancement with regard to AArchv7

- Register mapping between A32/T32 and A64
- Mapping of AArch64 System registers to the AArch32 System registers

ARMv8 Exceptions

- Four exception levels
- Exception Link Registers
- Register banking by exception level based on a new exception model
- Nesting on the same exception level
- Exception type and exception origin
- Syndrome registers used to provide a status information to the exception handler
- Exception return instruction

The ARMV8-A Security Model

- Security model when EL3 is using AArch64
- Trapping to EL3 using AArch64

Interprocessing

- Managing two types of processes: 64-bit and 32-bit, switching on an exception
- Non secure space organization

Second Day

Virtualization

- New hypervisor privilege level on non-secure side
- Re-entrant mode
- Hypervisor exception management, trapping
- Asynchronous exception routing and control

Instruction Pipeline

- Superscalar operation
- Predicted and non-predicted instructions
- Branch accelerators
- Invalidation and context switches
- Highlighting differences with Cortex-A57

Multicore

- Synchronization and semaphores
- Shareability memory attributes
- Operation of the global monitor
- Load acquire / Store release instruction pair
- Use of WFE and SEV instructions by spin-locks
- CLREXMON request and acknowledge signaling

Memory Accesses

- Mixed-endian support
- Program counter and stack pointer alignment
- Ordering requirements
- Page attributes : Normal or Device
- Shareability and access limitations on the data barrier operations
- Memory barriers

ARMv8 MMU Support

- LPAE enhancements to adapt to AArch64
- Supporting up to 48 bits of VA per TTBR
- Access permission checking
- Supporting up to 48 bits of IPA and PA spaces
- VMSAv8-64 address translation system
- Memory translation granule size
- Descriptor page table organization, descriptor format
- Hierarchical control of Secure or Non-secure memory accesses
- TLB preload instructions
- TLB maintenance instructions in A64
- Cortex-A73 TLB implementation

Third Day

Caches

- Cache hierarchy, Point of Unification, Point of Coherency
- Load non temporal instruction
- Instruction and Data cache maintenance instructions in A64
- Cortex-A73 L1 and L2 memory system
- L2 hardware cache flush
- L2 replacement algorithm selection

A64 New Instruction Set

- A64 assembly language, regular bit encoding structure
- Instruction aliases
- Branches, function call and return
- Conditional select instructions, avoiding branches
- Load Store instructions, addressing modes
- Arithmetic and logical instructions, CRC calculation instructions
- Instructions for accessing AArch32 Execution environment registers

ARM Architecture Procedure Call Standard 64-bit

- General register usage convention
- Stack pointer and frame pointer
- NEON / VFP register usage convention

NEON, VFP and Cryptographic Units

- New register banking for NEON and VFP
- Mapping of the SIMD and floating-point registers between the Execution states
- Vector formats in AArch64 state
- New SIMD instructions
- Cryptography software support through a new family of instructions

GICv4

- Generic Interrupt Controller CPU interface registers
- Interrupt virtualization
- Interrupt handling to support nesting

Fourth Day

Generic Timer

- System counter clock frequency
- Physical and virtual timer count registers
- Physical up-count comparison, down-count value and timer control registers
- Virtual up-count comparison, down-count value and timer control registers

Low Power States

- Wait for Interrupt and Wait for Event
- Cortex-A73 low power modes
- L2 Wait for Interrupt
- Processor dynamic retention
- Support for power management with multiple power domains
- Dormant mode
- Processor dynamic retention
- L2 RAMs dynamic retention
- Core Q-channel interface

ARMv8 Debug

- Self-hosted debug
- Debug state instructions
- Linked comparisons for Breakpoint/Watchpoint exception generation
- Software Step exceptions
- Routing debug exceptions
- External debug, cross-triggering
- Embedded Trace Macrocell architecture

Performance Monitor

- Per-function performance monitoring at EL0 level
- Effect of EL3 and EL2 on Performance Monitor
- Event filtering

Cortex-A73 Hardware Implementation

- Clocking
- Resets