



## RA5 - Cortex-A17 implementation

*This course covers the Cortex-A17 cluster*

### OBJECTIVES

- This course is split into 3 important parts:
  - Cortex-A17 architecture
  - Cortex-A17 software implementation and debug
  - Cortex-A17 hardware implementation.
- Introduction to Hypervisor new privilege mode is done at the beginning of this course.
- The consequences on address translation is then explained, introducing the 2-stage translation.
- Decoupling guest OS from hardware using traps to Hypervisor is studied.
- The course also details the new features of the Generic Interrupt Controller v2, explaining how physical interrupt requests can be virtualized.
- The course details the new approach regarding integrated timers / counters.
- AXI v4 new capabilities are highlighted with regard to AXI v3.
- Through sequences involving a Cortex-A17 and a Cortex-A7MP, the hardware coherency is studied, explaining how snoop requests can be forwarded by CCI-400 interconnect.
- Implementation of I/O MMU-400 is also covered.

A more detailed course description is available on request at [training@ac6-training.com](mailto:training@ac6-training.com)

### PREREQUISITES AND RELATED COURSES

- Knowledge of Cortex-A9.
- More than 12 correct answers to Cortex-A prerequisites questionnaire.
- Related courses:
  - VFP programming, reference [RC0 - VFP programmingcourse](#)
  - NEON programming, reference [RC1 - NEON-v7 programmingcourse](#)

### 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

## OVERVIEW OF CORTEX-A17

- Cortex-A17 architecture
- Organization of a SoC based on Cortex-A17
- AMBA4 coherent interconnect capabilities
- Inner Shareable vs Outer Shareable attribute
- I/O MMU
- 64-Byte cacheline size, integrated L2 cache
- VFPv4 and SIMDv2
- Highlighting differences between Cortex-A9 and Cortex-A17

## INSTRUCTION PIPELINE

- Global organization, dual issue capability
- Fetch / decode / rename / dispatch stages
- Loop mode
- Execution clusters
- Out-of-order execution
- Branch accelerators

## INTRODUCTION TO HYPERVISOR STATE

- Processor privilege levels state machine, user, guest OS, hypervisor
- Detailing the various operation modes (Bare-Metal, Hypervisor kernel and user task, Hypervisor with Guest partition)
- Asymmetric approach, no support for Virtualization of Secure state functionality
- SVC, HVC and SMC instructions
- Objective of the Hypervisor
- Hypervisor related instructions and registers
- List of registers that have to be saved / restored to be able to suspend / resume a guest partition
- Accessing banked registers or any Non-Secure mode while running in Hypervisor mode

## EXCEPTION MECHANISM

- Hypervisor vector table
- Utilization of Vector #5 to trap Guest partition events
- System Call into Hypervisor mode
- Asynchronous exceptions
- Virtual Interrupt and Abort bits control, IRQ, FIQ, external abort routing control
- Hypervisor exception return
- Taking exceptions into Hypervisor mode

## GENERIC INTERRUPT CONTROLLER (GICv2)

- Integration in a SoC based on Cortex-A17 and Cortex-A7
- Highlighting the new features with regard to Cortex-A9
- Steering interrupts to guest OS or Hypervisor
- Virtual CPU interface
- Split EOI functionality
- Deactivating an interrupt source from the Virtual CPU interface
- Front-end interface accessed by the Guest Kernel
- Back-end interface accessed by the Hypervisor

## **VIRTUALIZATION EXTENSIONS**

- New Intermediate Physical Address, 2-stage address translation
- Memory translation system
- Memory management when running in hypervisor mode
- Virtual Machine Identification
- Exposing the MMU to Other Masters, IO MMU
- Emulation support, trapping load and store and executing them in Hypervisor state
- Second-stage access permissions and attributes

## **LARGE PHYSICAL ADDRESS EXTENSIONS SPECIFICATION (LPAE)**

- Need to introduce support for a second stage of translation as part of the Virtualization Extensions
- New 3-level translation
- Level-1 table descriptor format
- Level-2 table descriptor format
- Attribute and Permission fields in the translation tables
- Improving the caching of translation entries by providing contiguous hints
- complete set of cache allocation hints
- Handling of the ASID in the LPAE
- New cache and TLB maintenance operations

## **MMU IMPLEMENTATION**

- TLB organization, L1-TLB, L2-TLB
- TLB match process
- Coherent table walk
- Determining the exact cause of aborts through status registers
- Behavior when MMU is disabled

## **OS SUPPORT – SYNCHRONIZATION OVERVIEW**

- Inter-Processor Interrupts
- Barriers
- Cluster ID
- Exclusive access monitor, implementing Boolean semaphores
- Global Monitor
- Spin-lock implementation
- Using events
- Indicating the effect of Multi Core on debug interfaces

## **LEVEL ONE SUBSYSTEM**

- Physically Indexed Physically Tagged caches
- Pseudo LRU replacement algorithm
- Speculative accesses
- Hit Under Miss, Miss under Miss
- Write streaming threshold definition
- L1 cache hardware prefetch
- MOESI data cacheline states

## **LEVEL TWO SUBSYSTEM**

- Cache organization
- Optional ECC protection
- L2 cache hardware flush
- ACE master interface

- ACMP master interface
- ACP slave interface
- By means of sequences involving a multi-core Cortex-A17 and external masters, understanding how snoop requests can be used to maintain coherency of data between caches and memory

## GENERIC TIMER

- ARM generic 64-bit timers for each processor
- Virtual time vs Physical time
- Effect of virtualization on these timers
- Event stream purpose
- Kernel event stream generation
- Hypervisor event stream generation
- Gray count timer distribution scheme

## PERFORMANCE MONITORING VIRTUALIZATION EXTENSIONS

- Hypervisor performance monitoring
- Guest OS performance monitoring
- Lazy switching of PMU state by a hypervisor
- Reducing the number of counters available to a Guest OS
- Fully virtualizing the PMU identity registers

## AMBA4

- AXI-4
  - Quality of Service signaling
  - Updated meaning of Read Allocate and Write Allocate
  - Transaction buffering
- AXI-4 stream protocol
  - Byte types, data, position, null
  - Byte stream
  - Sparse stream
  - Data merging, packing, and width conversion
- AXI-4 lite
  - Burst length of 1
  - No exclusive access support
- AXI Coherency Extension (ACE)
  - Shareability domains
  - Coherency model, cache states
  - Additional channel signals
  - New channels, snoop address, snoop response, snoop data
  - Studying through sequences how a load request and a store request will be handled whenever they are marked as outer shareable requests
  - Using ReadUnique, CleanUnique and MakeUnique requests
  - Distributed Virtual Memory (DVM)
  - DVM synchronization message
  - Selecting the coherency state machine: MESI or MOESI according to the capabilities of the interconnect
  - Snoop filtering
- Exported barriers
  - DMB / DSB inner shareable, outer shareable or system

## HARDWARE IMPLEMENTATION

- Clock domains
- Resets, power-on reset timing diagram
- Valid reset combinations
- Power domains

- Power-on reset sequence, soft reset sequence
- Power management, processor and L2 state retention
- Maintaining coherency while CPUs are in standby state
- Interface to the Power Management Unit
- Powering down a CPU
- External debug over power down

## **CCI-400 CACHE COHERENT INTERCONNECT**

- AMBA 4 snoop request transport
- Snoop connectivity and control
- Connecting 2 CPUs through CCI, managing coherency domains
- Example of Cortex-A7 dual core and Cortex-A17 dual core

## **CORESIGHT DEBUG**

- Program Trace Macrocell
- Cross Trigger Interface and Cross Trigger Matrix for multi-processor debugging
- Adding Virtual Machine ID in the criterion used to set a breakpoint / watchpoint
- Tracking VMID change in trace output