



RA3 - Cortex-A15 implementation

This course covers Cortex-A15 high-end ARM CPU

OBJECTIVES

- his course is split into 3 important parts:
 - Cortex-A15 architecture
 - Cortex-A15 software implementation and debug
 - Cortex-A15 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-A15 and a Cortex-A7, 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 formation@ac6-formation.com

PREREQUISITES AND RELATED COURSES

- Knowledge of Cortex-A9.
- More than 12 correct answers to Cortex-A prerequisites questionnaire.
- Related courses:
 - Programming with RVDS IDE, reference cours [RV0 - Programming with RVDS IDE](#)
 - VFP programming, reference cours [RC0 - VFP programming](#)
 - NEON programming, reference cours [RC1 - NEON-v7 programming](#)

Environnement du cours

- Cours théorique
 - Support de cours au format PDF (en anglais) et une version imprimée lors des sessions en présentiel
 - Cours dispensé via le système de visioconférence Teams (si à distance)
 - Le formateur répond aux questions des stagiaires en direct pendant la formation et fournit une assistance technique et pédagogique
- Au début de chaque demi-journée une période est réservée à une interaction avec les stagiaires pour s'assurer que le cours répond à leurs attentes et l'adapter si nécessaire

Audience visée

- Tout ingénieur ou technicien en systèmes embarqués possédant les prérequis ci-dessus.

Plan du cours

First day

OVERVIEW OF CORTEX-A15MP

- Cortex-A15 architecture
- Organization of a SoC based on Cortex-A15MP
- 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-A15

INSTRUCTION PIPELINE

- Global organization, triple issue capability
- Fetch / decode / rename / dispatch stages
- Loop mode
- Execution clusters
- Out-of-order execution, 40-entry dispatch queue
- 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-A15MP and Cortex-A7MP
- Highlighting the new features with regard to Cortex-A9MP
- 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

Second day

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

Third day

LEVEL ONE SUBSYSTEM

- Physically Indexed Physically Tagged caches
- LRU replacement algorithm, implementation with a 2-way cache
- Speculative accesses
- Hit Under Miss, Miss under Miss
- Write streaming threshold definition
- Uploading the contents of L1 caches through dedicated CP15 registers
- MESI data cacheline states

LEVEL TWO SUBSYSTEM

- Cache organization
- Strictly enforced inclusion property with L1 data caches, simplification of snooping
- Optional ECC / parity protection
- Impact of registers slices on performance
- L2 prefetch engine
- Table walk access prefetch
- ACE master interface
- ACP slave interface
- By means of sequences involving a multi-core Cortex-A15 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

Fourth day

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
- 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
- By means of sequences involving a multi-core Cortex-A15 and external masters, understanding how snoop requests can be used to maintain coherency of data between caches and memory
- Connecting 2 CPUs through CCI, managing coherency domains
- Example of Cortex-A7 dual core and Cortex-A15 dual core

CORESIGHT DEBUG

- Program Trace Macrocell
- Cross Trigger Interface and Criss 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