RA4 - Cortex-A7 implementation

This course covers Cortex-A7 ARM CPU

OBJECTIVES

- This course is split into 3 important parts:
  - Cortex-A7(MP) architecture
  - Cortex-A7(MP) software implementation and debug
  - Cortex-A7(MP) 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-A15MP 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

PREREQUISITES AND RELATED COURSES

- More than 12 correct answers to Cortex-A prerequisites questionnaire.
- Related courses:
  - Programming with RVDS IDE,reference RV0 - Programming with RVDS IDEcourse
  - 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

First day

OVERVIEW OF CORTEX-A7MP

- Cortex-A7 architecture
- Organization of a SoC based on Cortex-A7MP
- AMBA4 coherent interconnect capabilities
- I/O MMU
- 64-Byte cacheline size, integrated L2 cache
- VFPv4 and SIMDv2
- Supported instruction sets
- Highlighting differences between Cortex-A9 and Cortex-A7

INSTRUCTION PIPELINE

- Global organization, dual issue capability
- Fetch / decode / issue / writeback stages
- Data processing unit
- 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)
- Objective of the Hypervisor
- Support for interrupt nesting in Hypervisor mode
- Detecting VFP/Neon utilization by a Guest partition

EXCEPTION MECHANISM

- Hypervisor vector table
- Utilization of Vector #5 to trap Guest partition events
- Virtual Interrupt and Abort bits control, IRQ, FIQ, external abort routing control
- 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

Second day

VIRTUALIZATION EXTENSIONS

- New Intermediate Physical Address, 2-stage address translation
- Memory translation system
- Memory management when running in hypervisor mode
- Exposing the MMU to Other Masters, IO MMU

- Emulation support, trapping load and store and executing them in Hypervisor state
- Additional security facilities

LARGE PHYSICAL ADDRESS EXTENSIONS SPECIFICATION (LPAE)

- New 3-level system
- Hypervisor-level address translation
- Level-1 table descriptor format
- Level-2 table descriptor format
- Attribute and Permission fields in the translation tables
- Handling of the ASID in the LPAE
- New cache and TLB maintenance operations

MMU IMPLEMENTATION

- TLB organization, L1-TLB, L2-TLB
- Coherent table walk
- Tablewalk cache and IPA cache operation
- Determining the exact cause of aborts through status registers
- Behavior when MMU is disabled
- TLB maintenance operations

OS SUPPORT SYNCHRONIZATION OVERVIEW

- Inter-Processor Interrupts
- Barriers
- Cluster ID
- Exclusive access monitor, implementing Boolean semaphores
- Global monitor
- Spin-lock implementation
- Using events

Third day

LEVEL ONE SUBSYSTEM

- Cache organization, 2-way instruction cache, 4-way data cache
- Speculative accesses
- Hit Under Miss, Miss under Miss
- Read allocate mode
- Uploading the contents of L1 caches through dedicated CP15 registers
- MOESI data cacheline states
- Detailing cache maintenance operations

LEVEL TWO SUBSYSTEM

- Optional L2 Cache
- Read allocate mode
- ACE master interface
- By means of sequences involving a multi-core Cortex-A7 and external masters, understanding how snoop requests can be used to maintain coherency of data between caches and memory
- Synchronization primitives, the 3 levels of monitors

GENERIC TIMER

- ARM generic 64-bit timers for each processor
- Virtual time vs Physical time

- Event stream purpose
- Kernel event stream generation
- Hypervisor event stream generation

PERFORMANCE MONITORING VIRTUALIZATION EXTENSIONS

- Hypervisor performance monitoring
- Guest OS performance monitoring
- Reducing the number of counters available to a Guest OS
- Fully virtualizing the PMU identity registers

Fourth day

AMBA4

- AXI-4
- AXI-4 stream protocol
- AXI-4 lite
- AXI Coherency Extension (ACE)
- Exported barriers

HARDWARE IMPLEMENTATION

- Clock domains
- Resets, power-on reset timing diagram
- Power domains
- Power-on reset sequence, soft reset sequence
- Power management, WFI / WFE, dormant mode based on L2 memory
- 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
- ACE master interface
- 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 Cross Trigger Matrix for multi-processor debugging
- Adding Virtual Machine ID in the criterion used to set a breakpoint / watchpoint