RS1 - Cortex-A9 & Cortex-A5 software implementation

This course describes the architecture of Cortex-A5/A9 and provides coding guidelines

Objectives

- *This course aims to explain all low level characteristics of the Cortex-A9 that are required to develop efficient Kernel or application code.*
- *MMU operation under Linux is described.*
- *Spin-lock implementation in a multicore system is also detailed.*
- *Interaction between level 1 caches, level 2 cache and main memory is studied through sequences.*
- *The exception mechanism is explained, indicating how virtualization enables the support of several operating systems.*
- *An overview of the Coresight specification is provided prior to describing the debug related units.*
- *The operation of the Snoop Control Unit when supporting SMP is fully explained, particularly the utilization of cache tag mirrors, the advantage of connecting DMA channels to ACP and the sequences that have to be used to modify a page descriptor.*

Labs are run under RVDS.
A more detailed course description is available on request at training@ac6-training.com

Prerequisites and related courses

- *More than 12 correct answers to our 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

INTRODUCTION TO CORTEX-A9

- *Block diagram, 1 or 2 AXI master interfaces*
- *Cortex-A9 variants: single core vs multicore*
- *New memory-mapped registers in MPCore*
- *Configurable options: cache size, Jazelle, NEON, FPU, PTM and IEM*

ARM BASICS

- *States and modes*
- *Benefit of register banking*
- *Exception mechanism*
- *Purpose of CP15*

INSTRUCTION PIPELINE

- *Superscalar pipeline operation*
- *Branch prediction mechanism*
- *Guidelines for optimal performance*
- *Return stack*

TRUSTZONE

- *TrustZone conceptual view*
- *Secure to non secure permitted transitions*
- *Memory partitioning*
- *Interrupt management when there is a mix of secure and non-secure interrupt sources*
- *Boot sequence*

OS SUPPORT SYNCHRONIZATION OVERVIEW

- *Inter-Processor Interrupts*
- *Barriers*
- *Cluster ID*
- *Exclusive access monitor*
- *Spin-lock implementation*
- *Using events*

Second day

THUMB-2, THUMB-2EE AND ARM INSTRUCTION SETS (V7-A)

- *Data processing instructions*
- *Branch and control flow instructions*
- *Memory access instructions*
- *Exception generating instructions*
- *If&then conditional blocks*
- *Stack in operation*
- *Accessing special registers*
- *Interworking ARM and Thumb states*

- *Thumb-2EE extension for supporting interpreted languages*
- *Using handlers to manage NULL pointers and array index that are outside a programmable range*

MEMORY MANAGEMENT UNIT

- *MMU objectives*
- *Page sizes*
- *Page access permission, domain and page protection*
- *Page attributes, memory types*
- *Utilization of memory barrier instructions*
- *Format of the external page descriptor table*
- *Tablewalk*
- *Abort exception, on-demand page mechanism*
- *MMU maintenance operations*
- *Using a common page descriptor table in an SMP platform, maintaining coherency of multiple TLBs*

LEVEL 1 MEMORY SYSTEM

- *Cache organization*
- *Supported maintenance operations*
- *Write and allocate policies*
- *Data prefetching*
- *4-entry 64-bit merging store buffer*

PL310 LEVEL 2 CACHE

- *Understanding through sequences how cacheable information is copied from memory to level 1 and level 2 caches*
- *Transient operations, utilization of line buffers LFBs, LRBs, EBs and STBs*
- *Discarding a level 3 memory line load through merging writes into STBs*
- *Cache event monitoring*
- *Describing each maintenance operation*
- *Cache lockdown, implementation of a small memory by a boot program*
- *Interrupt management*

Third day

HARDWARE COHERENCY

- *Snooping basics*
- *Cache-to-cache transfers*
- *MOESI state machine*
- *Address filtering*
- *Understanding through sequences how data coherency is maintained between L2 memory and L1 caches*
- *Accelerator Coherency Port*

PERFORMANCE MONITOR

- *Event counting*
- *Debugging a multi-core system with the assistance of the PMU*

INTERRUPT CONTROLLER

- *Cortex-A9 exception managemen*
- *Interrupt groups: STI, PPI, SPI, LSPI*
- *Assigning a security level to each interrupt source (Secure or Non Secure)*
- *Prioritization of the interrupt sources*
- *Distribution of the interrupts to the Cortex-A9 cores*

- *Detailing the interrupt sequence*

CORESIGHT DEBUG UNITS

- *Benefits of CoreSight*
- *Invasive debug, non-invasive debug, taking into account the secure attribute*
- *APBv3 debug interface*
- *Connection to the Debug Access Port*
- *Debug facilities offered by Cortex-A9*
- *Process related breakpoint and watchpoint*
- *Program counter sampling*
- *Event catching*
- *PTM interface, connection to funnel*
- *Cross-Trigger Interface, debugging a multi-core SoC*

COMPILER HINTS AND TIPS

- *Placing code, data, stack and heap in the memory map, scatterloading*
- *Reset and initialization*
- *Placing a minimal vector table*
- *Further memory map considerations, 8-byte stack alignment in handlers*
- *Building and debugging an image*
- *Long branch veneers*
- *ARM compiler optimisations, tail-call optimization, inlining of functions*
- *Mixing C/C++ and assembly*
- *Coding with ARM compiler*
- *Unaligned accesses*
- *Local and global data issues, alignment of structures*
- *Further optimisations, linker feedback*