RR2 - Cortex-R7 implementation

This course covers the Cortex-R7MP ARM cores

Objectives

- This course is split into 3 important parts:
  - Cortex-R7 architecture
  - Cortex-R7 software implementation and debug
  - Cortex-R7 hardware implementation.
- Interaction between level 1 caches, TCM and main memory is studied through sequences.
- The course explains how to assign access permissions and attributes to regions by using the MPU.
- The exception mechanism is detailed, indicating how the GIC can contribute to reduce interrupt latency.
- Sequences involving memory, cache and external masters are used to explain the benefits of the ACP port.
- The course also details the hardware implementation and provides some guidelines to design a SoC based on a Cortex-R7.
- An overview of the Coresight specification is provided prior to describing the debug related units.

A more detailed course description is available on request at training@ac6-training.com

Prerequisites and related courses

- Basic knowledge of the ARM architecture.
- Assembly-level programming notions

Course material

- Printed training material is given to attendees during training.
- Precise and easy to use, it can be used as a reference afterwards.

Target Audience

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

Course Outline

First day

ARM Basics

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

Introduction to Cortex-R7

- Block diagram
- Slave and master AXI ports
- Highlighting the new features with regard to Cortex-R4/R5
- ARMv7-R architecture

- Operating modes
- Supported instruction sets
- Program Status register
- Exceptions
- System control coprocessor
- Configurable options
- Implementing two CPUs
  - Cache coherency using the SCU
  - Accelerated Coherency Port
  - Redundant CPU vs Dual CPU
  - Split/Lock configuration

Cache Coherency

- Hardware coherency
  - SCU implementation
  - The MESI and MOESI protocols
- ACP interface, providing hardware coherency for DMA accesses
- PMU related events

Instruction Pipeline

- Prefetch unit
- Studying how instructions are processed step by step
- Instruction cycle timings and interlock behavior
- Dynamic branch prediction mechanism: global history buffer
- Guidelines for optimal performance
- Data Processing Unit
- Multiple issuing
- Global History Buffer
- Return stack
- Instruction Memory Barrier
- Prefetch queue flush
- PMU related events

Memory Types

- Memory types, restriction regarding load / store multiple
- Device and normal memory ordering
- Memory type access restrictions
- Access order
- Memory barriers, self-modifying code

Memory Protection Unit

- Memory protection overview, ARM v7 PMSA
- Default memory map
- Cortex-R7 MPU and bus faults
- Fault status and address registers
- Region overview, memory type and access control, sub-regions
- Region overlapping
- Setting up the MPU

Second day

OS support – Synchronization Overview

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

Exception management

- Low Interrupt Latency: abandoning load / store instructions in progress
- Configuring the state in which exceptions are handled: endian mode, instruction set
- Nested interrupt management
- Configuring the FIQ as non-maskable
- Abort exception, fault handling
- Determining the cause of the fault through CP15 status registers
- Precise vs imprecise faults

Generic Interrupt Controller (GICv1)

- Integration in a SoC based on Cortex-R7
- Cortex-R7 exception management
- Interrupt virtualization
- Integrated timer and watchdog unit in MPCore
- Interrupt groups: SGI, PPI, SPI, LSPI
- Legacy mode management for IRQ and FIQ
- Prioritization of the interrupt sources
- Distribution of the interrupts to the Cortex-R7 cores
- Detailing the interrupt sequence
- Spurious interrupt

Level 1 Memory System

- Cache basics: organization, replacement algorithm, write policies
- Cache organization
- Write with allocate policy
- Tag RAM and Data RAM organization
- Debugging when caches are active
- Parity / ECC protection, handling cache parity / ECC errors
- Understanding transient cache line load / store: linefill buffers, eviction buffer
- Cache maintenance operations
- Tightly Coupled Memories, address decoding
- ITCM and DTCM configuration
- Accessing the TCMs from the AXI slave interface
- ECC protection, TCM internal error detection and correction
- Preloading TCMs with ECC
- Using TCMs from reset
- Store buffer, merging data
- L1 caches software read for debug purposes
- PMU related events

Third day

Performance Monitoring Unit

- Event counting
- Selecting the event to be counted for the 3 counters
- Related interrupts
- Debugging a multi-core system with the assistance of the PMU
- Use of the event bus and counters

AXI Protocol

- Topology: direct connection, multi-master, multi-layer
- Separate address/control and data phases
- AXI channels, channel handshake
- Support for unaligned data transfers
- Transaction ordering, out of order transaction completion
- Read and write burst timing diagrams
- ECC management
- Write merging example
- Sideband signals

APB (Advance Peripheral Bus)

- Second-level address decoding
- Pinout
- Read timing diagram
- Write timing diagram
- APB3.0 new features

Cortex-R7 Level-2 Interface

- AXI Master interfaces
  - Main interface attributes
  - Optional second master interface
  - Identifying virtual masters
- AXI Peripheral interface
  - Peripheral interfaces port attributes
  - Identifiers for AXI peripheral port accesses
- Optional ACP port
- AXI slave interface
  - Slave interface attributes
  - Enabling or disabling AXI slave accesses
- Slave APB debug interface

Hardware Implementation

- Clock domains
- Reset domains, power-on reset and debug reset
- Power control, dynamic power management
- Separate debug and core power domains
- Clock gating
- Maintaining caches and TCM powered while turning off the pipeline: dormant mode
- Power mode interaction with ACP
- Wait For Interrupt architecture
- Debugging the processor while powered down

Low Power Modes

- Voltage domains
- Run mode, standby mode, dormant mode
- Studying the sequence required to enter and exit dormant mode
- Communication to the power management controller
- Standby and wait for event signals, implementation in a multi-core system

Coresight Debug Units Overview

- Benefits of CoreSight
- Invasive debug, non-invasive debug
- APBv3 debug interface
- Connection to the Debug Access Port
- Debug facilities offered by Cortex-R7
- Process related breakpoint and watchpoint
- Program counter sampling
- Event catching
- Debug Communication Channel
- ETM interface, connection to funnel
- Debugging while the processor is in shutdown or dormant mode
- Debug registers description
- Miscellaneous debug signals
- Cross-Trigger Interface, debugging a multi-core SoC
- Debugging systems with energy management capabilities