RR0 - Cortex-R4 implementation

This course covers the Cortex-R4 ARM core

Objectives

- This course is split into 3 important parts:
  - Cortex-R4 architecture
  - Cortex-R4 software implementation and debug
  - Cortex-R4 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 VIC port can contribute to reduce interrupt latency.
- The course also details the hardware implementation and provides some guidelines to design a SoC based on Cortex-R4.
- An overview of the Coresight specification is provided prior to describing the debug related units.

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

Prerequisites

- Knowledge of ARM7/9.
- This course does not include chapters on low level programming.
  - ACSYS offers a large set of tutorials to become familiar with RVDS, assembly level programming, compiler hints and tips.
- More than 12 correct answers to Cortex-R prerequisites questionnaire.

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

ARM BASICS

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

INTRODUCTION TO CORTEX-R4

- Block diagram
- ARMv7-R architecture
- Supported instruction sets
- Exceptions
- System control coprocessor
- Configurable options

INSTRUCTION PIPELINE

- Prefetch unit
- Instruction cycle timing
- Dynamic branch prediction mechanism
- Data Processing Unit
- Dual issue conditions
- Return stack
- Instruction Memory Barrier

MEMORY TYPES

- Device and normal memory ordering
- Memory type access restrictions
- Access order
- Memory barriers, self-modifying code

MEMORY PROTECTION UNIT

- ARM v7 PMSA
- Cortex-R4 MPU and bus faults
- Region overview, memory type and access control, sub-regions
- Region overlapping
- Setting up the MPU

EXCEPTION MANAGEMENT

- Low Interrupt Latency
- Primecell VICs
- VIC basic signal timing
- Interrupt priority and masking
- Abort exception
- Precise vs imprecise faults

Second day

LEVEL 1 MEMORY SYSTEM

- Cache basics
- Write with allocate policy
- Debugging when caches are active
- Accessing the cache RAM from AXI slave interface
- Tightly Coupled Memories
- ECC/parity protection
- Store buffer, merging data
- L1 caches software read for debug purposes

AXI PROTOCOL

- PL301 AXI interconnect
- Separate address/control and data phases
- AXI channels, channel handshake
- Support for unaligned data transfers
- Cortex-R4 external memory interface, ID encoding

HARDWARE IMPLEMENTATION

- Clock domains, CLKIN, FREECLKIN and PCLKDBG
- Reset domains, power-on reset and debug reset
- Power control, dynamic power management
- Wait For Interrupt architecture
- Debugging the processor while powered down

Third day

LEVEL 2 MEMORY SYSTEM

- AXI master interface
- Controlling an external cache
- AXI transaction splitting
- AXI slave interface
- Using the AXI slave interface to perform built-in self tests
- Understanding the error recovery mechanisms
- Exclusive accesses
- Local monitor

APB - ADVANCED PERIPHERAL BUS

- Pinout
- Read timing diagram
- Write timing diagram
- APB3.0 new features

PERFORMANCE MONITOR

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

LOW POWER MODES

- Voltage domains
- Run mode, standby mode, dormant mode
- Studying the sequence required to enter and exit dormant mode
- Standby and wait for event signals

CORESIGHT DEBUG UNITS

- Invasive debug, non-invasive debug
- APBv3 debug interface
- Debug facilities offered by Cortex-R4
- Process related breakpoint and watchpoint
- Program counter sampling

- Event catching
- Debug Communication Channel
- ETM interface, connection to funnel
- Cross-Trigger Interface, debugging a multi-core SoC

APB - ADVANCED PERIPHERAL BUS

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

DEBUG UNIT

- Performance monitor, event counting
- Coresight specification overview
- CP14 and memory-mapped registers
- Embedded core debug
- Invasive debug
- Debug exception
- Debug Communication Channel
- External debug interface
- Understanding how the Debug unit, the Embedded Trace Macrocell and the Cross-Triggering Interface interact