
ARM CORTEX-R4 SYSTEM DESIGN

Ref : 004837A

Duration : 4 days

OBJECTIVES

- This course takes an **in depth** look at the considerations you will need to take into account when designing a system containing either an ARM7TDMI family or ARM9TDMI family processor core. Information on the latest generation of ARM processor cores, is also included.
- It is aimed at :
 - Software engineers who not only want to obtain details of how to write software to run on the ARM, but also wish to obtain an understanding of hardware design issues
 - Hardware engineers who need to understand how to design ARM based systems, but also wish to obtain an understanding of the issues of writing software to run on that system

RELATED COURSES

- Formation 1 (référence)
- Formation 2 (référence)

PARTNERS

- This training course is approved by ARM

PREREQUISITES

- A basic understanding of microprocessors and microcontrollers is recommended
- A basic understanding of digital logic or hardware / ASIC design issues would be useful but not essential
- A basic understanding of assembler or C programming would be useful but not essential
- A basic awareness ARM cores is useful but not essential
- Note that the Cortex-R4 implements some features also supported by other ARM cores, particularly AXI bus and VIC ports interfaces (described in ARM1176 course) and Thumb-2 instruction set (described in Cortex-A8 course)
- According to the knowledge of attendees, these parts can be removed

PRACTICAL LABS

- For on-site courses, labs can be run under the following environments : CodeWarrior/ADS/AXD, Eclipse/RVDS, Keil µVision, GNU/Lauterbach simulator, or IAR Workbench
- For open courses, labs are run under Eclipse/RVDS



Contact

Tel : +33 (0)5 62 13 52 32
Fax : +33 (0)5 61 06 72 60
training@mvd-fpga.com

Course also available
customized

Next sessions, see : <http://www.mvd-fpga.com/en/formationsCalend.html>

TOPICS

First day

INTRODUCTION TO CORTEX-R4

- Block diagram
- Highlighting the new features with regard to other ARM cores
- ARMv7-R architecture
- Operating modes
- ARM instruction set
- Thumb-2 instruction set
- Program Status register
- Exceptions
- System control coprocessor
- Configurable options

THUMB-2 INSTRUCTION SET

- Introduction
- General points on syntax
- Data processing instructions
- Branch and control flow instructions
- Memory access instructions
- Exception generating instructions
- If...then conditional blocks
- Stack in operation
- Exclusive load and store instructions
- Accessing special registers
- Coprocessor instructions
- Memory barriers and synchronization
- Interworking ARM and Thumb states

- Demonstration of assembly sequences aimed to understand this new instruction set

VFPv3 FLOATING POINT UNIT

- Floating point number encoding (normalized, tiny, zero, infinite, NAN)
- Overview of VFPv3-D16 architecture
- General purpose registers, FPU views of the register bank
- Compliance with IEEE754 standard
- Exception management
- NaN handling
 - Demonstration of floating point calculations generated from C language

COMPILER HINTS AND TIPS

- Automatic optimization
- Instruction scheduling
- Tail-call optimization
- Parameter passing
- Array and structure access
- Loop termination
- Inline assembler
- Stack usage
- Global data layout
- Highlighting some optimisations through practical labs, for instance tail-call optimization

Second day

EMBEDDED SOFTWARE DEVELOPMENT

- ROM/RAM remapping
- Exception vector table
- Reset handler
- Initialization : stack pointers, code and data areas
- C library initialization
- Scatterloading
- Linker placement rules
- Long branch veneers
- C library functionality
- Placing the stack and heap

INSTRUCTION PIPELINE

- Prefetch unit
- Studying how instructions are processed step by step
- Instruction cycle timing
- Dynamic branch prediction mechanism : global history buffer
- Guidelines for optimal performance
- Data Processing Unit
- Dual issue conditions
- 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
- Cortex-R4 MPU and bus faults
- Fault status and address registers
- Region overview, memory type and access control, sub-regions
- Region overlapping
- Setting up the MPU

EXCEPTION MANAGEMENT

- Low Interrupt Latency : abandoning load / store instructions in progress
- Configuring the state in which exceptions are handled : endian mode, instruction set
- Configuring the FIQ as non-maskable
- Primecell VICs
- Reducing interrupt latency through automatic vector generation
- VIC basic signal timing
- Connectivity : daisy-chained VIC
- Interrupt priority and masking
- Abort exception, fault handling
- Determining the cause of the fault through CP15 status registers
- Precise vs imprecise faults

Third day

HARDWARE IMPLEMENTATION

- Clock domains, CLKIN, FREECLKIN and PCLKDBG
- Using clock enable to determine the ratio between input clock and operation clock
- Reset domains, power-on reset and debug reset
- Power control, dynamic power management
- Wait For Interrupt architecture
- Debugging the processor while powered down

AXI PROTOCOL

- Topology : direct connection, multi-master, multi-layer
- PL300 AXI interconnect
- 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
- Atomic transactions

LEVEL 1 MEMORY SYSTEM

- Cache basics : organization, replacement algorithm, write policies
- Cache organization
- Write with allocate policy
- Debugging when caches are active
- Parity / ECC protection
- Understanding transient cache line load / store : linefill buffers, eviction buffer
- Accessing the cache RAM from AXI slave interface
- Tightly Coupled Memories, address decoding, enabling on reset
- ECC/parity protection
- Interleaving BTCM accesses initiated by core and AXI DMA connected to AXI slave interface
- Store buffer, merging data
- L1 caches software read for debug purposes
- PMU related events

Fourth day

LEVEL 2 MEMORY SYSTEM

- AXI master interface, write issuing capability, read issuing capability
- AXI transaction identifiers
- Controlling an external cache
- Restrictions on AXI transfers
- Determining the number and type of AXI transactions according to memory attributes and instruction type
- AXI transaction splitting
- AXI slave interface, write issuing capability, read issuing capability
- Enabling or disabling AXI slave accesses
- Using chip-select to distinguish A-TCM, B-TCM, I-Cache and D-Cache accesses
- Using the AXI slave interface to perform built-in self tests
- Understanding the error recovery mechanisms
- Exclusive accesses, swap instructions, internal exclusive monitor, requirement of an external exclusive monitor when implementing multiple cores

APB - ADVANCED PERIPHERAL BUS

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

DEBUG UNIT

- Performance monitor, event counting
- Related interrupts, event bus
- Coresight specification overview
- CP14 and memory-mapped registers, utilization of an APB slave interface
- APB port access permissions
- Embedded core debug
- Invasive debug : breakpoints and watchpoints
- Vector catch

- Debug exception
 - Debug Communication Channel
 - External debug interface
 - Understanding how the Debug unit, the Embedded Trace Macrocell and the Cross-Triggering Interface interact
 - Debugging systems with energy management capabilities
-

DOCUMENTATION

Training manuals will be given to attendees during training in print.