FA3 - i.MX51 Implementation + LTIB

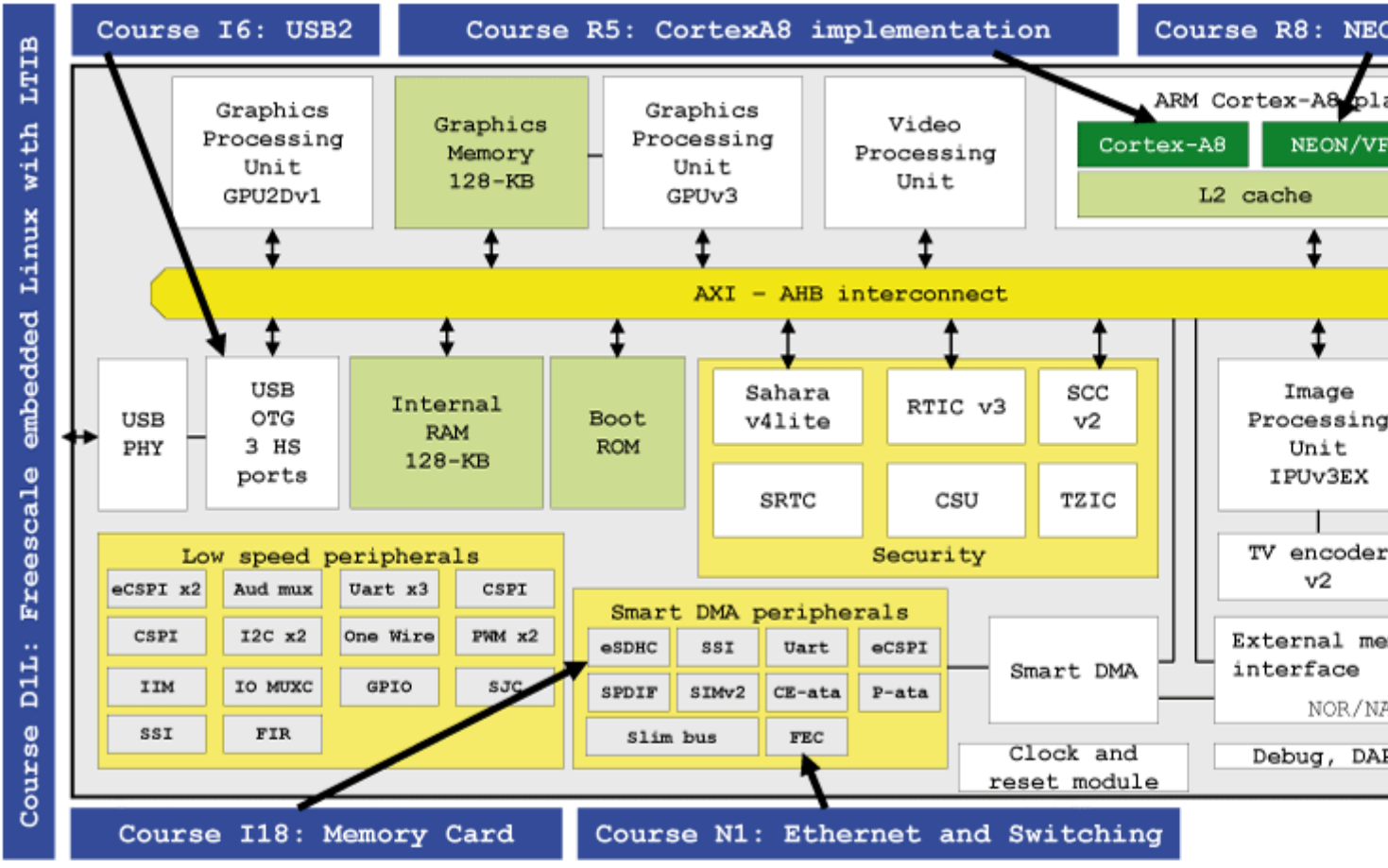
This course describes the i.MX51 multimedia processor and Linux Target Image Builder tool

Course objectives

- The course details the hardware implementation of the MCIMX51 microcontroller.
- The course focuses on the boot sequence, the clocking and the power management strategies.
- The course explains all parameters that affect the performance of the system in order to easily perform the final tuning.
- The multiple complex units involved in multimedia stream management are covered in depth.
- An overview of the Cortex-A8 core helps to understand issues caused by cache and MMU.
- The course ends with practical labs explaining how to generate a Linux image as well as a Root File System, by using a tool called LTIB [Linux Target Image Builder]

- Products and services offered by ACSYS:
  - ACSYS has developed FFTs (floating-point and fixed-point) optimized for ARM cores, based on SIMD instructions supported by the Cortex-A8.
  - Contact formation@ac6-formation.comto obtain informations about the performance of these FFTs.
  - ACSYS is able to assist the customer by providing consultancies. Typical expertises are done during board bringup, hardware schematics review, software debugging, performance tuning.
  - ACSYS has also an expertise in programming the SDMA, a simple OS-agnostic driver has been developed to explain how to manage scripts.

Related courses



Prerequisites

- Knowledge of ARM architecture is recommended
- Knowledge of Linux basics is recommended

Documentation

- Training manuals will be given to attendees during training both in pdf and in print. Precise and easy to use, those notes can be used as a reference afterwards.

Target Audience

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

Course Outline

Architecture of MCIMX51

- Clarifying the internal data paths : AXI interconnect, AHB bus, peripheral buses
- Highlighting the purpose of the 2 central interconnect units : MAX and M4IF
- Organization of a board based on MCIMX51
- Mapping

The ARM Cortex/A8 Core - Overview

- Operating modes : user, system, super, IRQ, FIQ, undef and abort
- ARM vs Thumb-2 instruction sets, interworking
- Access to memory-mapped locations, addressing modes
- Stack management
- Branch instructions, implementation of C call and return statements
- Level1 cache operation
- Level2 cache operation
- Memory management unit, TLB
- C-to-Assembly interface
- Exception mechanism, handler table

Reset and Clocking

- Clock distribution
- DVFS support
- Power Gating Controller
- Low power modes, wake-up detector
- Global reset vs warm reset
- System boot mode selection
- eFUSE configuration

System Control

- GPIO module
- General Purpose Input interrupt request capability

The Cortex/A8 Platform

- MAX parameterizing
- ARM Vector Interrupt Controller

- Integrated timers EPIT, GPT, WDT

Debug Architecture

- Introduction to CoreSight, DAP features
- System Secure Controller SJC
- Embedded Trace Macrocell
- Cross Triggering Interfaces

Smart DMA Controller

- Mapping DMA requests to channels
- Channel priority definition
- Scheduler
- Instruction description
- PCU states
- Context switching
- Reference clocks and low power modes
- Debug support
- Profiling unit

Accessing External Memories

- Description of the Master Arbitration and Buffering [MAB] unit
- Description of the M4IF arbitration [M3A] unit
- Introduction to DDR2/LPDDR SDRAM
- Enhanced DDR2 SDRAM controller
- NAND flash controller, boot from flash

System Security

- Security Controller
  - Protecting information and data from unauthorized access
  - A dedicated AES cryptographic engine
  - High Assurance Boot
- SAHARA4 security coprocessor
  - Random number generator
  - Encryption / decryption sequences
  - Restricted access to potentially sensitive information
  - ARM TrustZone support
- Run-Time Integrity Checker
  - SHA-1 and SHA-256 message authentication
  - Segmented data gathering
  - One-time hash mode vs continuous hash mode
- IC Identification Module

Standard Parrallel Interfaces

- ATA controller
  - Pinout
  - PIO mode
  - Ultra DMA mode
- Enhanced SDHC
  - Interface to SD, MMC, SDIO and CE-ATA cards
  - Transfer protocol, single block, multiple block read and write
  - Internal and external DMA capabilities
  - Error management

Video Processing Units

- Video Processing Unit
  - Codec hardware
  - Encoding pipeline
  - Video Codec processing buffer requirement
- Image Processing Unit v3
  - Video acquisition
  - Image Signal Processor, processing captured images
  - Processing chain description
  - Display processor, processing chain
  - Video de-interlacer
  - Image converter
  - Image rotator
  - Display port
- Graphics Processing Unit 2D
  - 2D bitmap graphics
  - Vector graphics
  - Connection to DMA controller
- Graphics Processing Unit 3D
  - Sophisticated shader support
  - Graphics core
  - Graphics memory
  - Pixel blender
  - Integrated MMU
- TV encoder
  - Supported TV standards, SD/HD modes
  - TV signal processor
  - Cable detection circuit

Audio Related Interfaces

- SSI interfaces
  - Connection of Codecs or DSPs
  - I2S mode
  - AC97 support
- Digital audio multiplexor
  - Connecting host interfaces to peripheral interfaces
  - Internal network mode
- SPDIF transmitter
  - Selecting the clock
  - Transmit FIFO operation

Communication Controllers

- 1-wire interface
- Configurable SPI, enhanced CSPI
  - SPI protocol basics
  - Transfer sequence
- High Speed I2C and I2C interfaces
  - I2C protocol basics
  - Transfer sequence
- Fast Infrared Interface [FIRI]
  - MIR packet structure, MIR modulation
  - FIR packet structure, FIR modulation
- UART
  - Individual baud rate generators

    - Flow control
- USB
    - Explaining what is OTG
    - The 3 USB ports
    - High-speed operation
    - EHCI support
    - ULPI bypass mode
- Fast Ethernet Controller [FEC]
    - Ethernet basics
    - Incoming frame filtering mechanisms, hash tables
    - Flow control in Full Duplex mode
    - VLAN support
- SIM
    - Introduction to IEC / ISO 7816
    - Transferring packets

Generating the Linux Kernel Image

- Introducing the tools required to generate the kernel image
- What is required on the host before installing LTIB
- Common package selection screen
- Common target system configuration screen
- Building a complete BSP with the default configurations
- Creating a Root File Systems image
- Re-configuring the kernel under LTIB
- Selecting user-space packages
- Setup the bootloader arguments to use the exported RFS
- Debugging Uboot and the kernel by using Trace32
- Command line options
- Adding a new package
- Other deployment methods
- Creating a new package and integrating it into LTIB

Exercise: Several labs will help explain the usage of LTIB