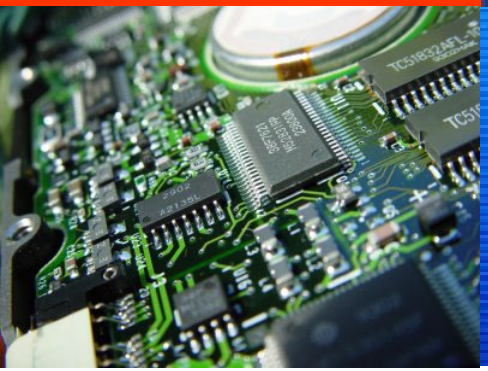


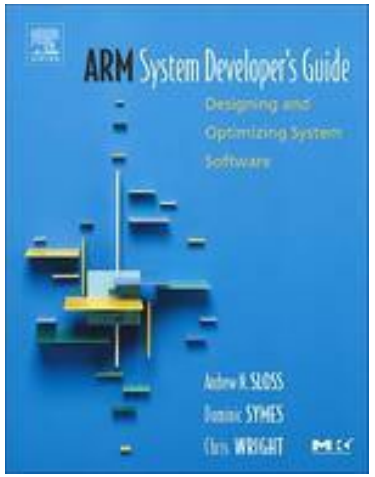
SoC Design : Lecture 1

Prof. Jong-Myon Kim

울산대학교
UNIVERSITY OF ULSAN

Course Information

- Instructor
  - Jongmyon Kim (jongmyon.kim@gmail.com)
- Office: Engineering Building 1, Room 7-308
- Class Hour : Tuesday 4, 5, 6 period (1pm-4pm)
- Course website:
  - http://ulms.ulsan.ac.kr
  - Constantly updated, check it out regularly
- Textbook
  - Slides & Lectures
  - [Option] Andrew N. Sloss, *ARM System Developer's Guide*, Elsevier, 2004.
- Other teaching materials
  - Some key papers available in class meetings and course web



Text Book

Weekly Plan

- Week 1: Introduction to ARM SoC
- Week 2: Introduction to SoC Processor
- Week 3: ARM Assembly Language Programming
- Week 4: ARM Organization and Implementation
- Week 5: ARM Instruction Set
- Week 6: Architectural Support for High-Level Languages
- Week 7: Thumb Instruction Set
- Week 8: *Mid-Term Exam*
- Week 9: Architectural Support for System Development
- Week 10: ARM Processor Cores
- Week 11: Memory Hierarchy
- Week 12: Architectural Support for Operating Systems
- Week 13: ARM CPU Cores
- Week 14: Embedded ARM Applications
- Week 15: *No Class*
- Week 16: *Final Exam*

Grading Policy

- Exams: 90%
  - Mid-Term Exam: 45%
  - Final Exam: 45%
- Class Participation: 10%

Course Objective

- Understanding Fundamental Concept for SoC
- Understanding SoC
- Introduction to SoC Design Process
- Understanding Fundamental Concept for SoC Platform

Contents

- RISC Design Philosophy
- ARM Design Philosophy
- Embedded System Hardware
- Embedded System Software

RISC Design Philosophy

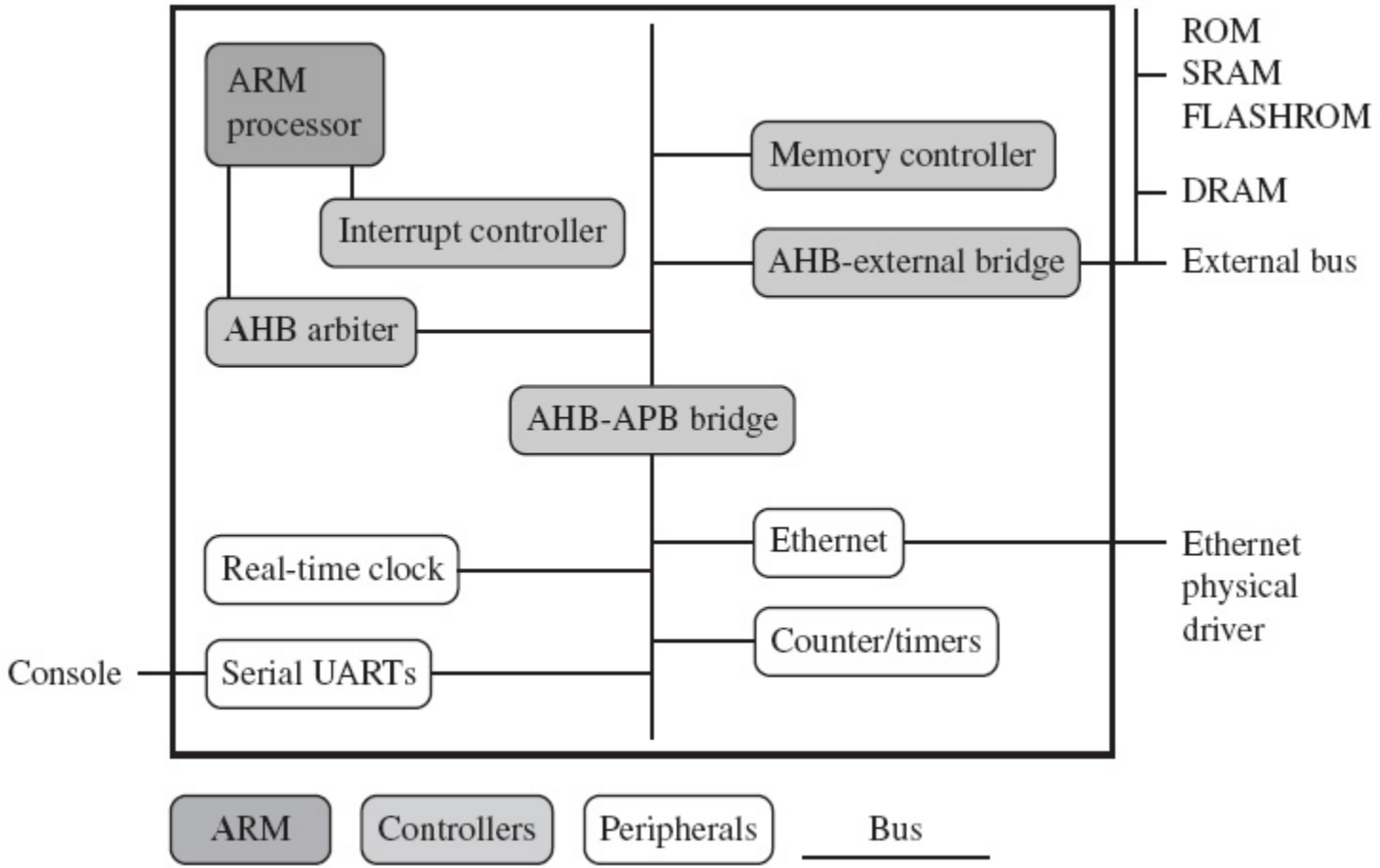
- Instructions
  - A reduced number of instruction classes
    - Execute in a single cycle
  - A fixed size: 32-bit
    - Allow the pipeline to fetch future instructions before decoding the current instruction
- Pipelines
  - Instructions can be decoded in one pipeline stage in parallel
- Registers
  - A large general-purpose register set
- Load-store architecture
  - Transfer data between the register bank and external memory

ARM Design Philosophy

- Variable cycle execution for some instructions
  - LDM/STM
- Inline barrel shifter leading to more complex instructions
  - A hardware component that preprocesses one of the input registers before it is used by an instruction
- Thumb 16-bit instruction set
  - Permit the ARM core to execute either 16- and 32-bit instructions
- Conditional execution
  - Execute when a specific condition has been satisfied
- Enhanced DSP instructions
  - Support fast 16x16-bit multiplier operations and saturation

System-on-Chip Hardware

- An example of an ARM-based embedded device, a microcontroller



ARM Processor

- Controls the embedded device
- ARM Core + (MMU/MPU) + (Caches)

Controllers

- Memory Controller
  - Connect different types of memory to the processor bus
- Interrupt Controller
  - When a peripheral or device requires attention, it raises an interrupt to the processor

Peripherals

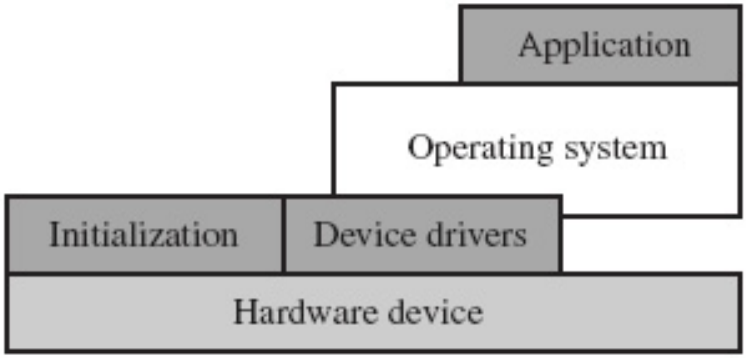
- Provide all the input-output capability external to the chip
- Memory mapped
  - the programming interface is a set of memory-addressed registers
- Examples
  - Serial communication devices
  - 802.11 wireless device
  - Real-Time Clock

Bus

- Communicates between different parts of the device
- AMBA (Advanced Microcontroller Bus Architecture)
  - ASB (ARM System Bus)
  - APB (ARM Peripheral Bus)
  - AHB (ARM High Performance Bus)
    - Multi-layer AHB
    - AHB-lite

ARM Embedded Software

- Initialization Code (Boot Code)
  - Set up the minimum parts of the board
- Device Drivers
  - provides a standard interface to peripherals
- Operating System
  - Control applications and manage hardware system resources
- Application
  - Perform one of the tasks required for a device



Initialization Code

- Configures the hardware to known state
  - Configures the memory controller and processor caches
  - initializes some devices
- Phases
  - Initial Hardware Configuration
  - Diagnostics
  - Booting

Device Driver

- provides a standard interface to peripherals

Operating System

- Provides a common programming environment for the use of hardware resources and infrastructure
- ARM processors support over 50 operating systems.
- 2 main categories
  - Real-time Operating System (RTOS)
  - Platform Operating System

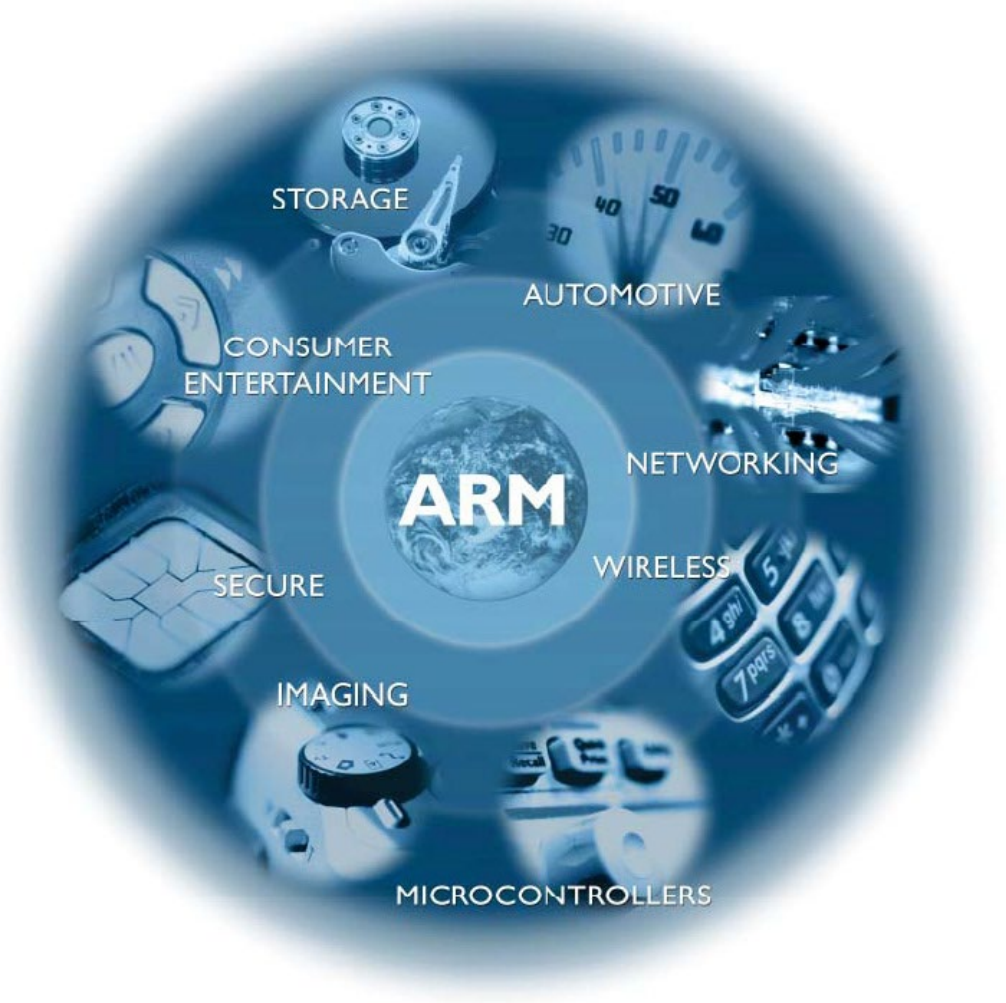
Real-Time Operating Systems

- Provide guaranteed response times to events
- Hard real-time application
  - Requires a guaranteed response to work
- Soft real-time application
  - Requires a good response time
- Ex) VxWorks, pSOS, ThreadX, NucleausOS

Platform Operating Systems

- Requires a MMU to manage large, non real-time applications
- Ex) Linux, WinCE

Applications



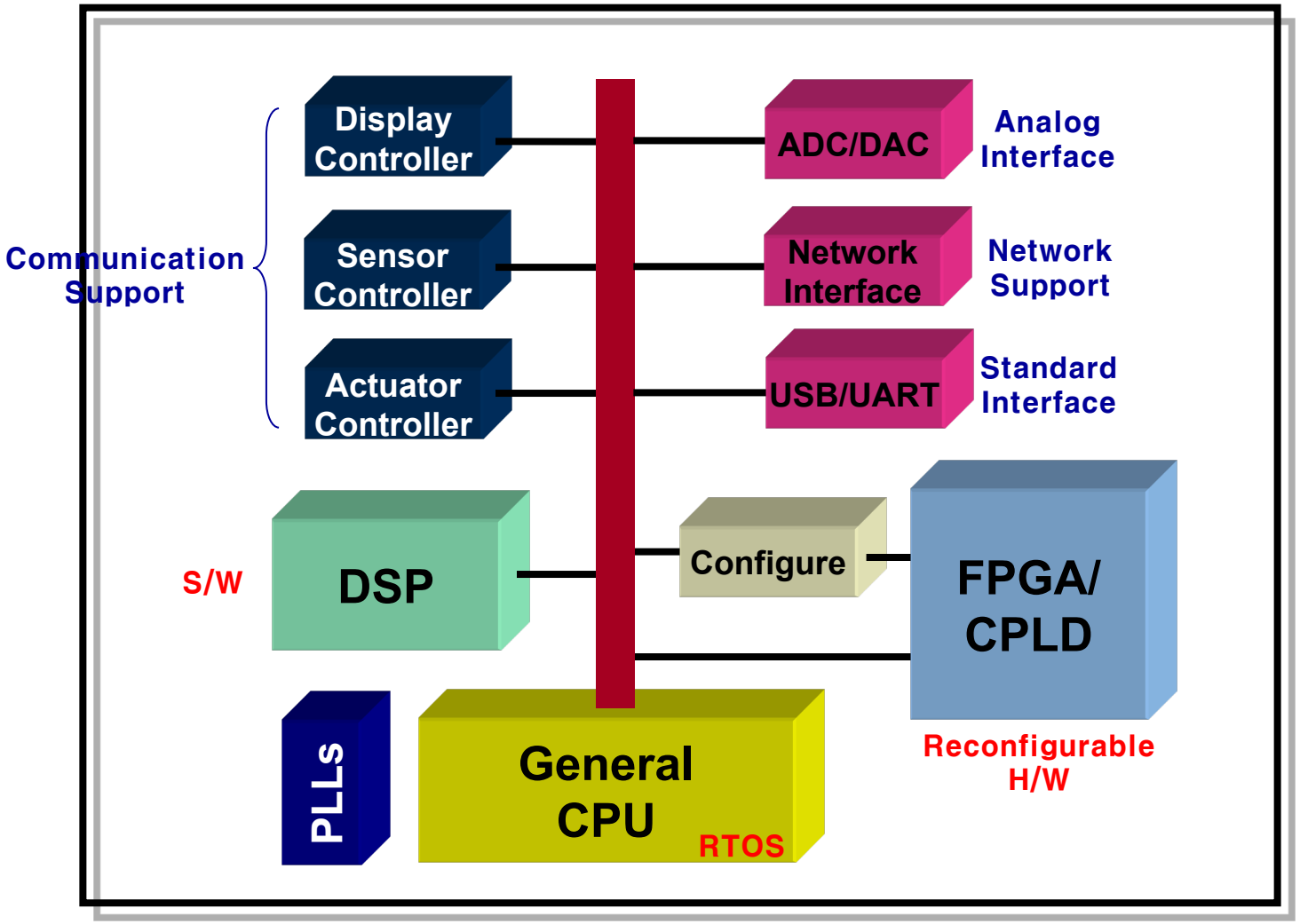
SoC Platform Architecture

- Embedded Processor
  - ARM : ARM922T, ARM926EJ-S, ARM1136
  - Altera : Excalibur (ARM922T, FPGA)
  - Motorola : PowerPC
  - XilinX : Virtex II Pro (IBM405GP, FPGA)
  - MIPS : MIPS4K
- Reconfigurable FPGA
  - XilinX Virtex II, Spartan
  - Altera Stratix, Cyclone, APEX, Excalibur
- Memory Block
  - SDRAM, DDRRAM
  - Flash Memory

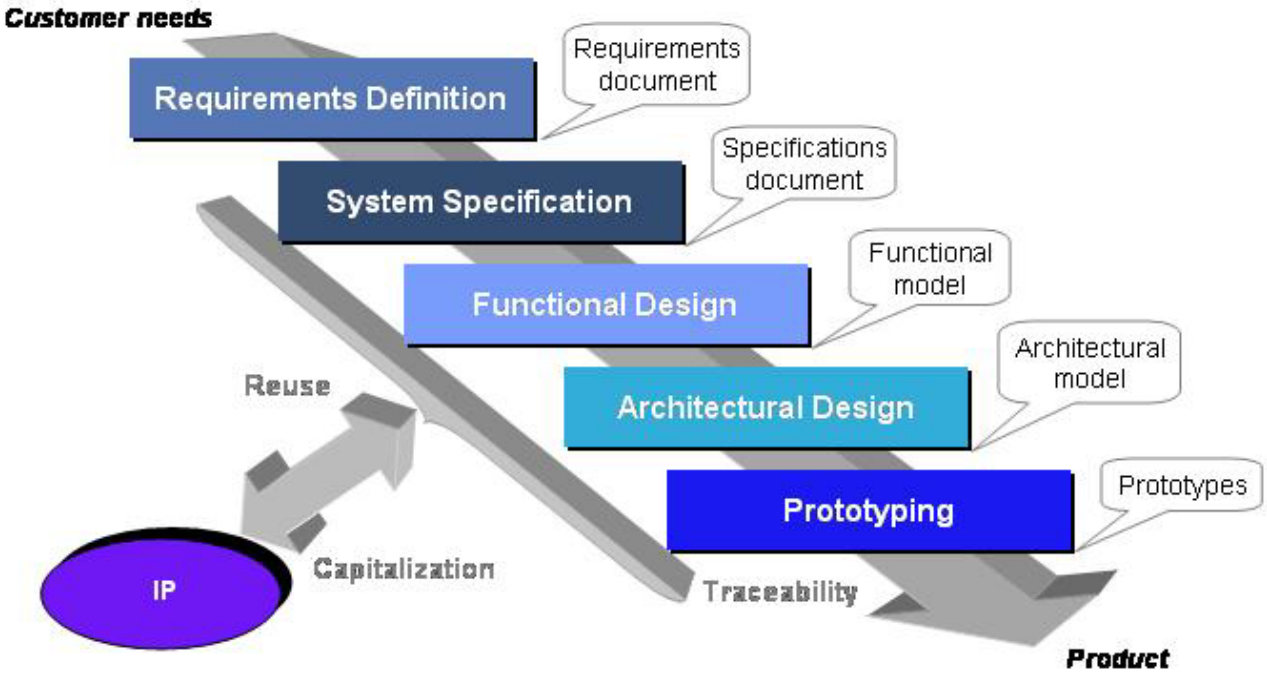
SoC Platform Architecture

- All the peripheral functions required for a basic SoC
  - UART, Ethernet 10/100M, 1394, USB2.0
  - PCI, PC104, PCMCIA, CF+, Cardbus
  - DMA, Interrupt Request,
  - TFT LCD, Text LCD, FND
  - Key PAD, LED, PB Switch
- IP
  - MP3, MPEG4, H.264, MJPEG, JPEG, Memory Controller, PCI
  - USB Controller, UART Controller etc
- Bus Architecture
  - AMBA Spec. 2.0
- Operating System
  - Linux, Nucleus RTOS, Vxworks, Qplus

SoC Platform Architecture



5 Design Step for SoC



5 Design Step for SoC

- Requirements definition
- System specification
- Functional design
- Architectural design
- Prototyping

Design steps	Purpose	Inputs	Outputs
1 . Requirements definition	Understanding all interested parties' needs and documenting these needs as written definitions and descriptions.	Customer needs	Requirements document
2 . System specification	Representing a purely external view of the system. Completely modeling the behavior of the system within its environment and listing all non-functional constraints.	Requirements document	Specifications document
3. Functional design	Finding a suitable internal logical (or functional) and technology-independent representation of the system from an application-oriented viewpoint.	Specifications document	Functional model
4 . Architectural design	Defining the detailed system's physical (or hardware) architecture and the organization of the software application on each programmable processor.	Functional model Performances/cost constraints	Architectural model
5.Prototyping	Developing an operational system prototype in terms of hardware and software implementations.	Architectural model Technological & economical constraints	Prototype