arm

Day 4: Silicon to System design

Silicon for Embedded Systems – An Overview

Workshop on

“VLSI to System design : Silicon-to-end application approach”

Desikan Srinivasan
August 2023

AICTE | arm Education | ST life.augmented

Supported by

cadence

Speaker Introduction

- **Desikan Srinivasan**
  - **Director of Engineering, CPU Group, Arm Bangalore**

  - **25+ years in VLSI industry**
  - **Design verification**

Supported by

Disclaimer

Arm is committed to making the language we use inclusive, meaningful, and respectful. Our goal is to remove and replace non-inclusive language from our vocabulary to reflect our values and represent our global ecosystem.

Arm is working actively with our partners, standards bodies, and the wider ecosystem to adopt a consistent approach to the use of inclusive language and to eradicate and replace offensive terms. We recognise that this will take time. This course contains references to non-inclusive language; it will be updated with newer terms as those terms are agreed and ratified with the wider community.

Contact us at education@arm.com with questions or comments about this course. You can also report non-inclusive and offensive terminology usage in Arm content at terms@arm.com.

Module Syllabus

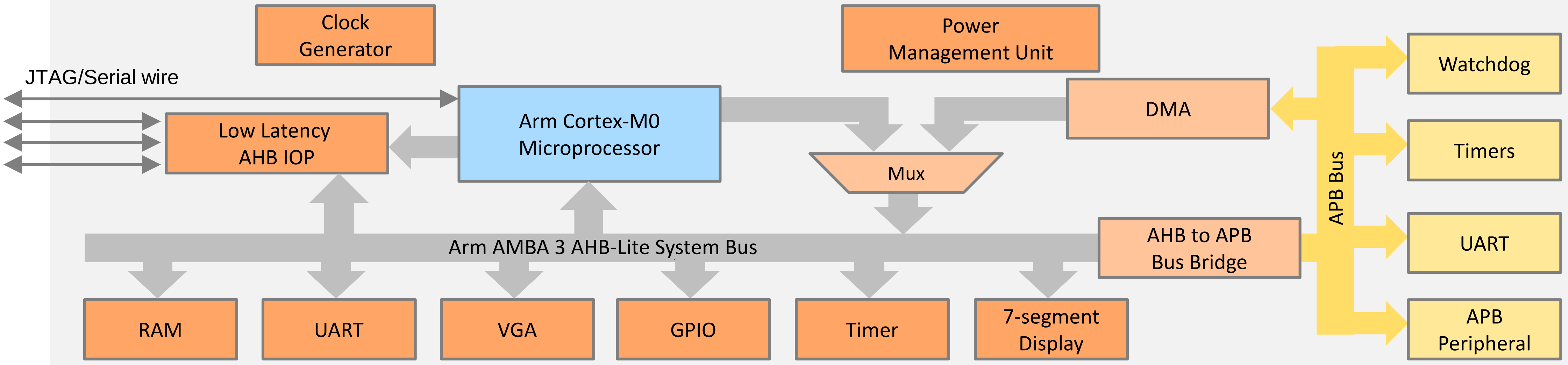
- Building a System on Chip

- Introduction to Embedded Systems
  - CPUs vs. MCUs vs. Embedded Systems
  - Examples of Embedded Systems
  - Options for Building Embedded Systems

- Arm processor-based Design.
  - Cortex-M series processors
  - Salient design features
  - Example MCU using Cortex-M processor

System on Chip (SOC)

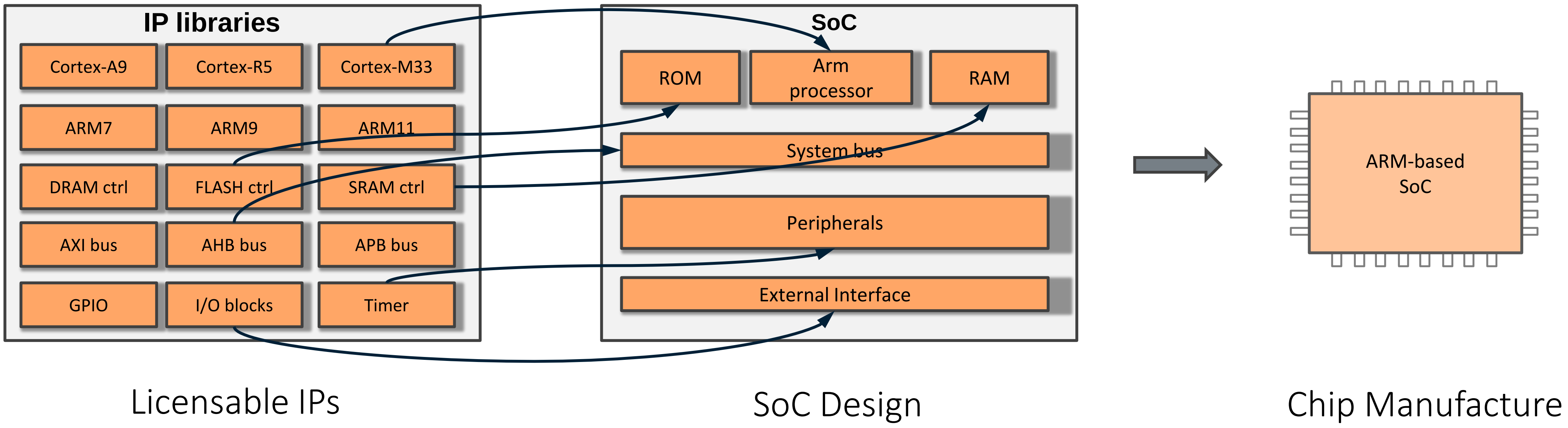
Example Arm-based SoC



An example of an Arm-based SoC

How to design an ARM-based SoC

- Select a set of IP cores from Arm and/or other third-party IP vendors
- Integrate IP cores into a single chip design
- Give design to semiconductor foundries for chip fabrication

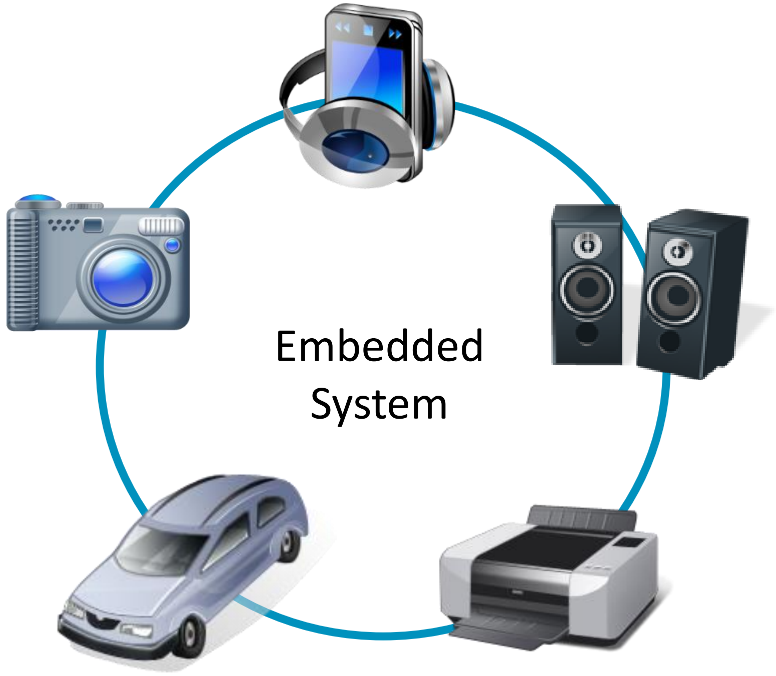


Embedded Systems

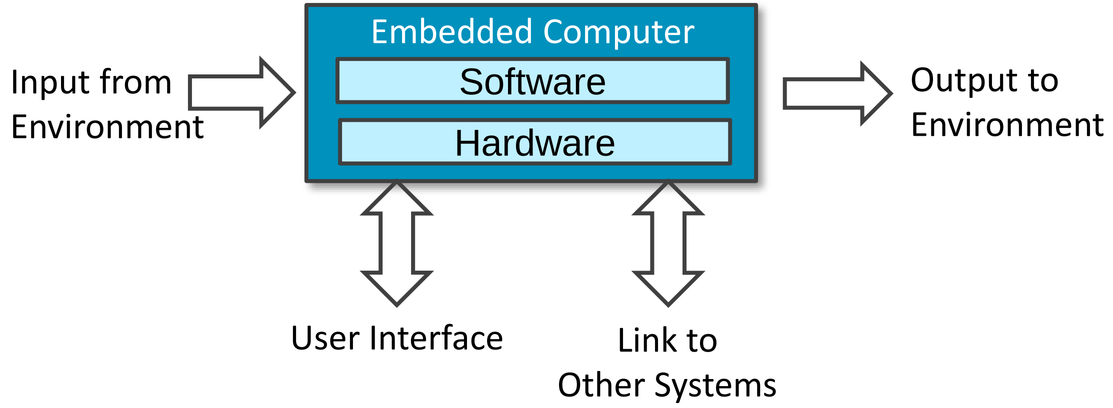
Introduction to Embedded Systems

- What is an embedded system?
  - Application-specific computer system
  - Interacting with its environment
  - Build into a larger system
  - Often with real-time computing constraints

- What is the motivation for building an embedded system?
  - Better performance
  - More functions and features
  - Lower cost e.g. through automation
  - More dependable
  - Lower power



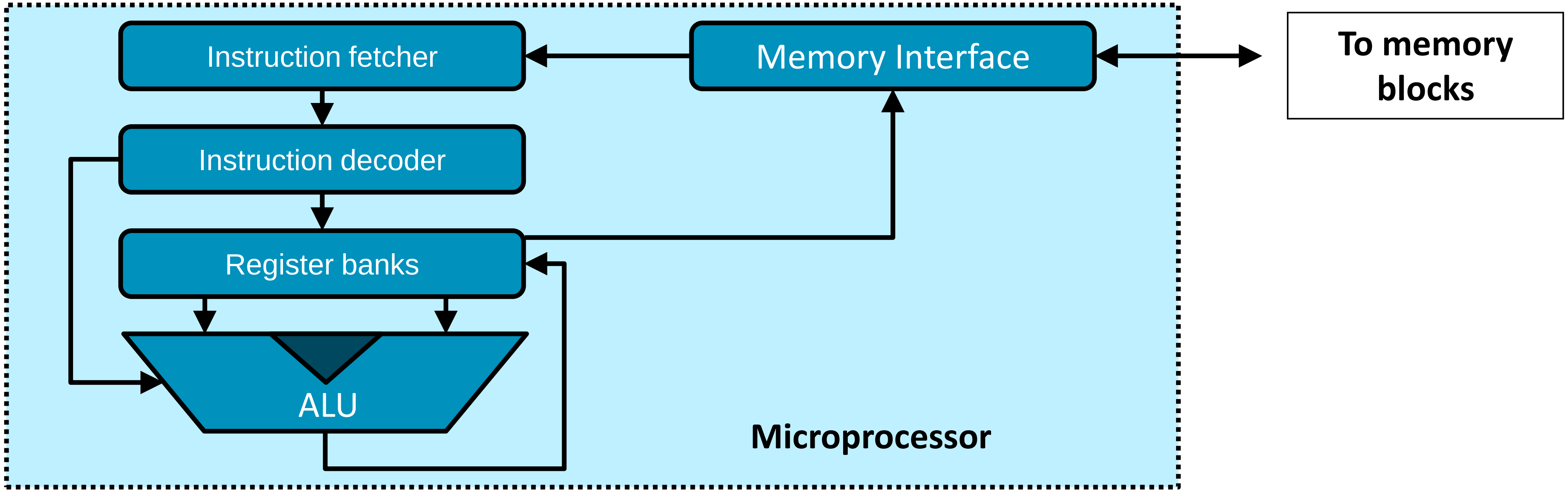
Macroscopic view on a device level



Microscopic view on a functionality level

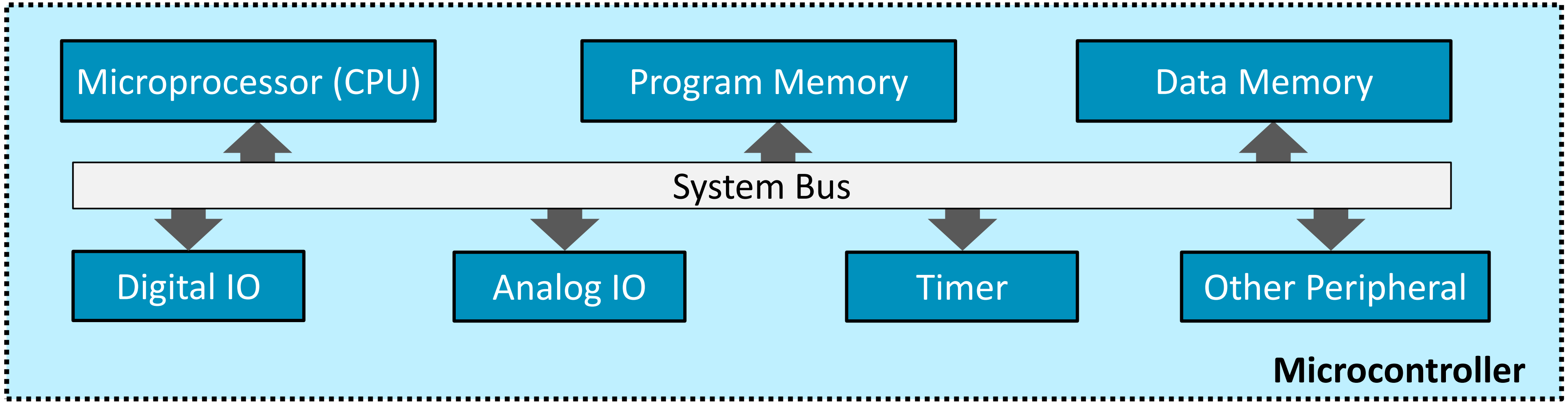
From a Processor to an Embedded System

- Microprocessor (central processing unit, CPU)
  - Typically defined as a single processor core that supports at least instruction fetching, decoding, and executing
  - Normally can be used for general-purpose computing, but needs to be supported with memory and Input/Outputs (IOs)



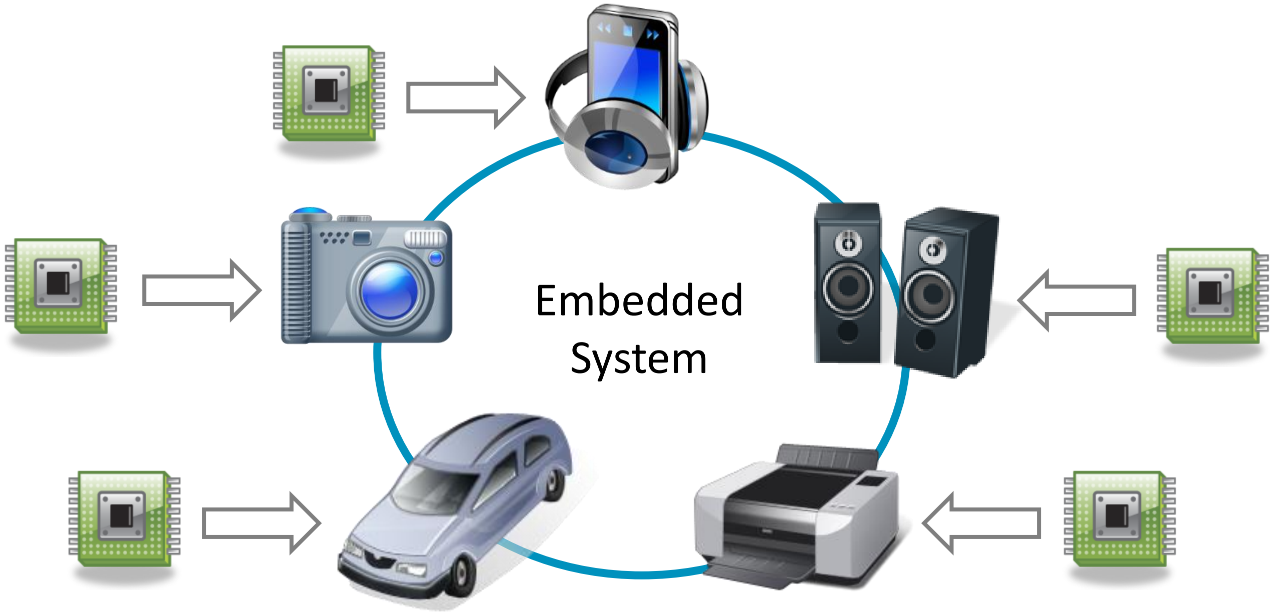
From a Processor to an Embedded System

- Microcontroller (microcontroller unit, MCU)
  - Typically has a single processor core
  - Has memory blocks, digital IOs, analog IOs, and other basic peripherals
  - Typically used for basic control purpose, such as embedded applications



From a Processor to an Embedded System

- Embedded system
  - Typically implemented using MCUs
  - Often integrated into a larger mechanical or electrical system
  - May have real-time constraints



Example Embedded System: Bike Computer

Functions:
- Speed measurement
- Distance measurement

Constraints:
- Size
- Cost
- Power and energy
- Weight

Inputs:
- Wheel rotation indicator
- Mode key

Output:
- Liquid crystal display

Use low-performance microcontroller:
- 9-bit, 10 MIPS

Input:
Wheel rotation
Mode key

Output:
Display speed and distance

Example: Gasoline Automobile Engine Control Unit

Functions:
- Fuel injection
- Air intake setting
- Spark timing
- Exhaust gas circulation
- Electronic throttle control
- Knock control

Many inputs and outputs:
- Discrete sensors and actuators
- Network interface to rest of car

Constraints:
- Reliability in harsh environment
- Cost
- Size

Use high-performance microcontroller:
- E.g. 32-bit, 3 MB flash memory, 50-300 MHz

arm

Introduction to the Internet of Things (IoT)

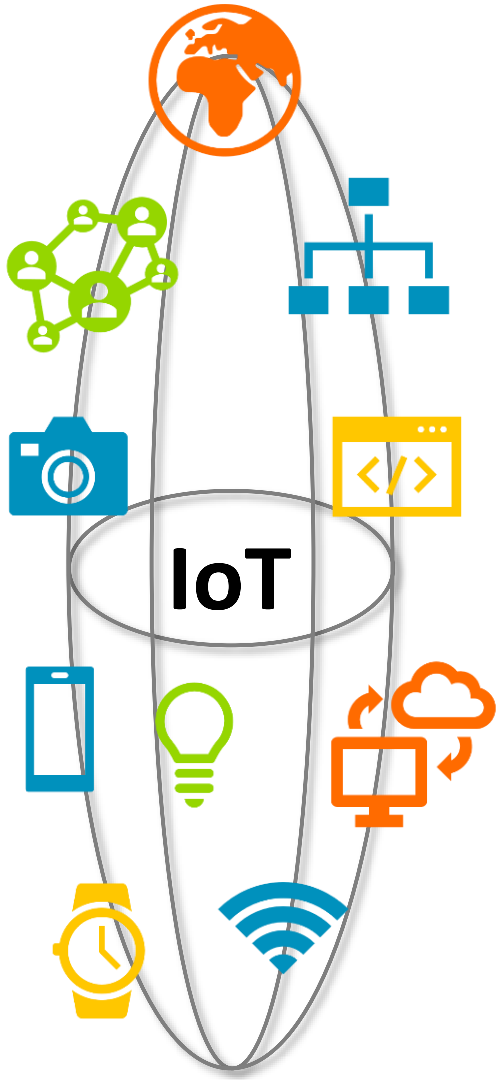
- IoT generally refers to a world in which a large range of objects are addressable via a network

Why IoT?

- Items can have more functionality and become more intelligent
- Items can be managed more easily
- More information becomes available

Objects can include:

- Smart buildings and home appliances
  - Fridges, TVs, cookers
- Civil engineering structures
  - Bridges, railways
- Wearable devices
  - Smart watches, glasses
- Medical devices
  - Smart inhaler, embedded pills



Options for Building Embedded Systems

	Implementation	Design Cost	Unit Cost	Upgrades & Bug Fixes	Size	Weight	Power	System Speed
Dedicated Hardware	Discrete logic	low	mid	hard	large	high	?	very fast
	ASIC	high ($500K/ mask set)	very low	hard	tiny – 1 die	very low	low	extremely fast
	Programmable logic – FPGA, PLD	low to mid	mid	easy	small	low	medium to high	very fast
Software Running on Generic Hardware	Microprocessor + memory + peripherals	low to mid	mid	easy	small to medium	low to moderate	medium	moderate
	Microcontroller (int. memory & peripherals)	low	mid to low	easy	small	low	medium	slow to moderate
	Embedded PC	low	high	easy	medium	moderate to high	medium to high	fast

Microcontroller based embedded system

arm

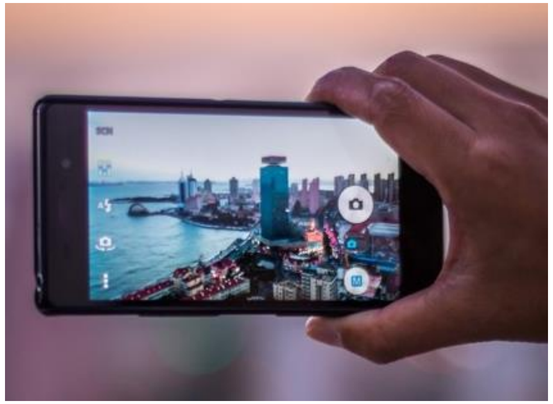
ARM Processor based Silicon

Arm architecture and processors

A family of RISC-based processor architecture

- Architecture well-known for its power efficiency
- Widely used in mobile and IoT devices, e.g., smartphones, connected cars, wearables
- Designed and licensed by Arm to a wide ecosystem of partners
- Arm does not manufacture, but it licenses designs to semiconductor companies, who add their own intellectual property (IP) on top of Arm's IP
- Arm also offers IP other than for processors, such as physical IPs, interconnect IPs, graphics cores, and development tools.

Arm Architecture Profiles



Cortex-A
Highest performance

Designed for high-level operating systems

Cortex-R
Faster responsiveness

Designed for high performance, hard real-time applications

Cortex-M
Smallest/lowest power

Designed for discrete processing and microcontrollers

SecurCore
Tamper resistant

Designed for physical security

Cortex-M Processor Value Proposition

Ease of use	Lower software cost	Pure C target CMSIS support Broad tools and OS support Binary compatible roadmap
Energy efficient	Lower energy cost	Sleep mode support Wake-up Interrupt Controller Low power implementation
Reduced system cost	Lower silicon cost	Thumb-2 code density Area-optimized designs CoreSight debug support

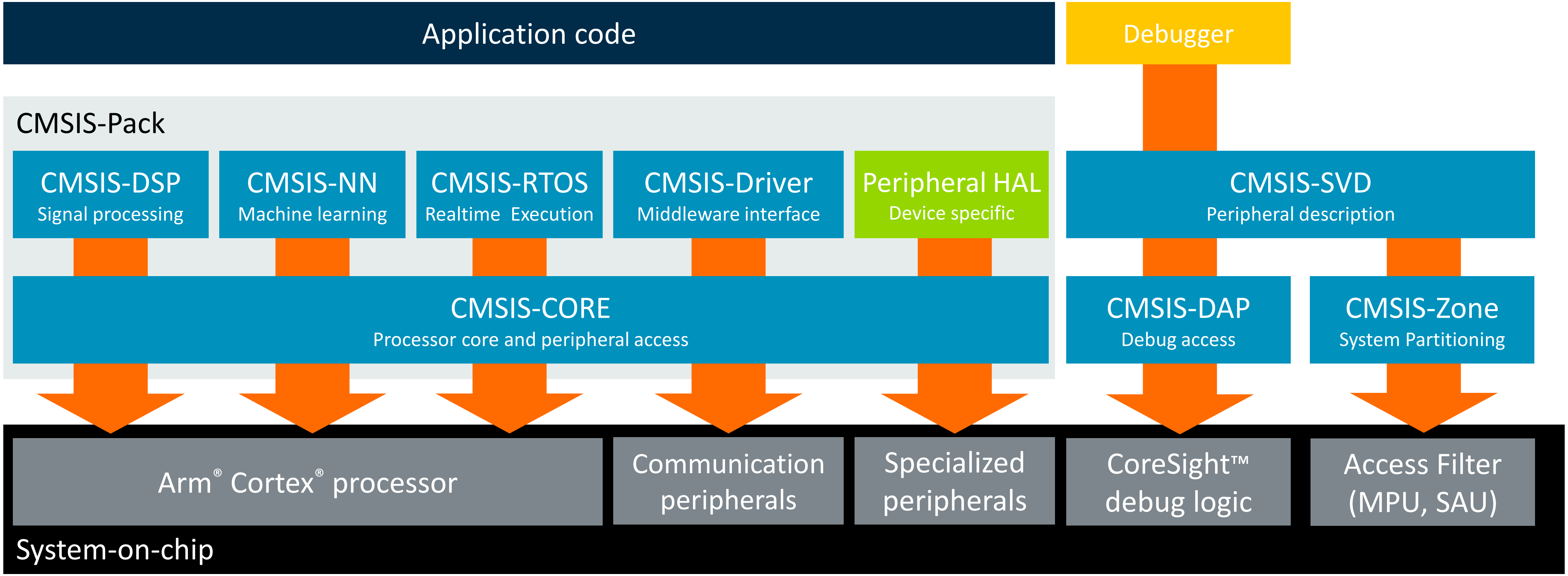
arm

CMSIS Simplifies Connecting Applications To The MCU

- Consistent software framework for Arm Cortex-M and Cortex-A5/A7/A9 based systems

CMSIS 5

- Consistent software framework for Arm Cortex-M and Cortex-A5/A7/A9 based systems



Cortex-M processors serving all applications

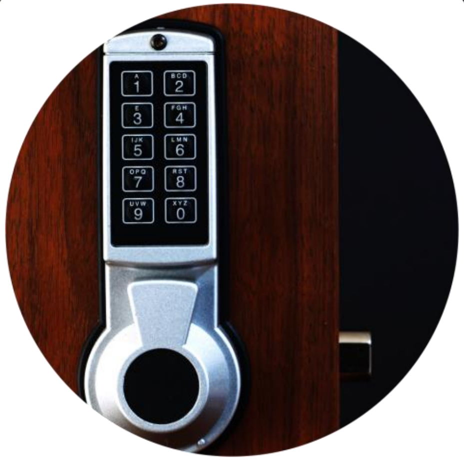
Energy grid
Automotive
Environmental
Home automation
Healthcare
Enterprise
Retail

Smart city
Wearables
Farming
Identity & tracking
VR / AR
Building automation
Connected clothing

Robotics
Sensor
Industrial
IoT
Smart lighting
Smart watch
Space

arm

IoT Will Be Everywhere



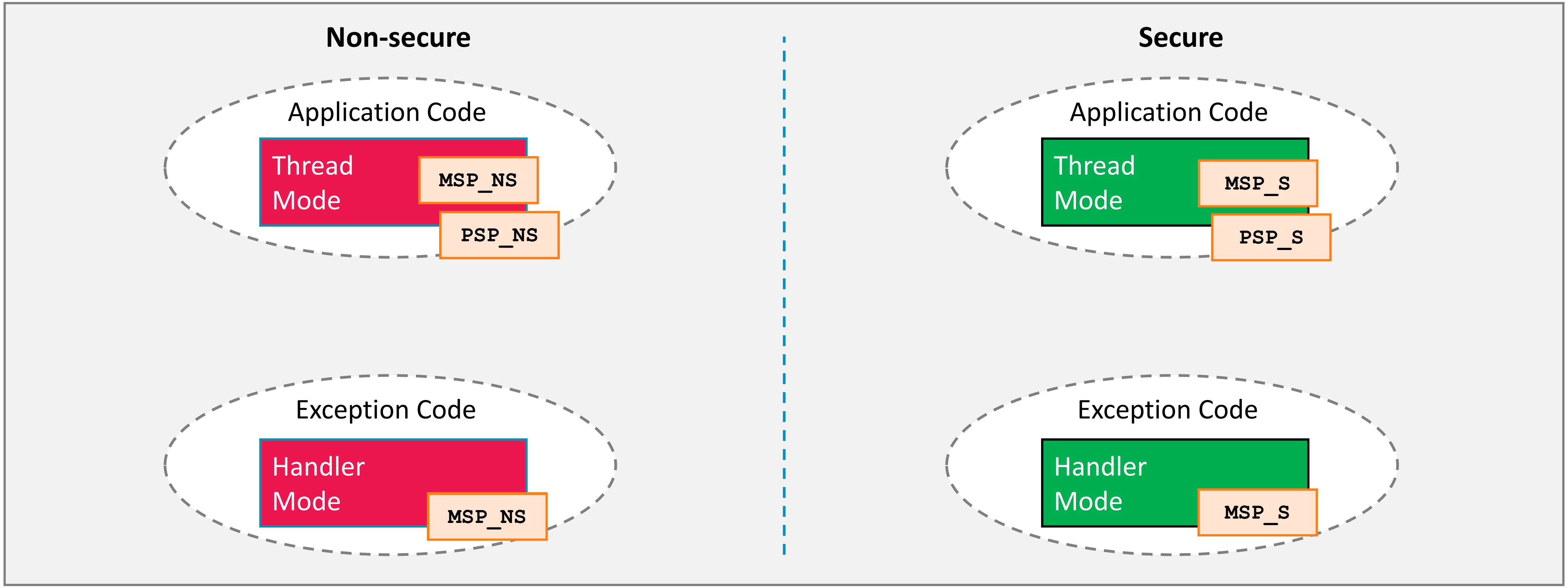
How do we design in robust end-to-end security?

http://www.flickr.com/photos/jurvetson
/7408464122/in/photostream/

Introduction to TrustZone for Armv8-M

- Armv8-M architecture includes optional Security Extension
  - Branded as Arm TrustZone for Armv8-M
- Similar in concept to TrustZone for Armv8-A
  - Implementation is optimized for microcontrollers
- System may be partitioned between secure and non-secure software
- Secure software is highly trusted
  - Has access to more system resources
  - Protected from access by non-trusted code
- To protect the secure software the security extensions provide:
  - Isolated Secure memory for code and data
  - Secure execution state to run Secure code

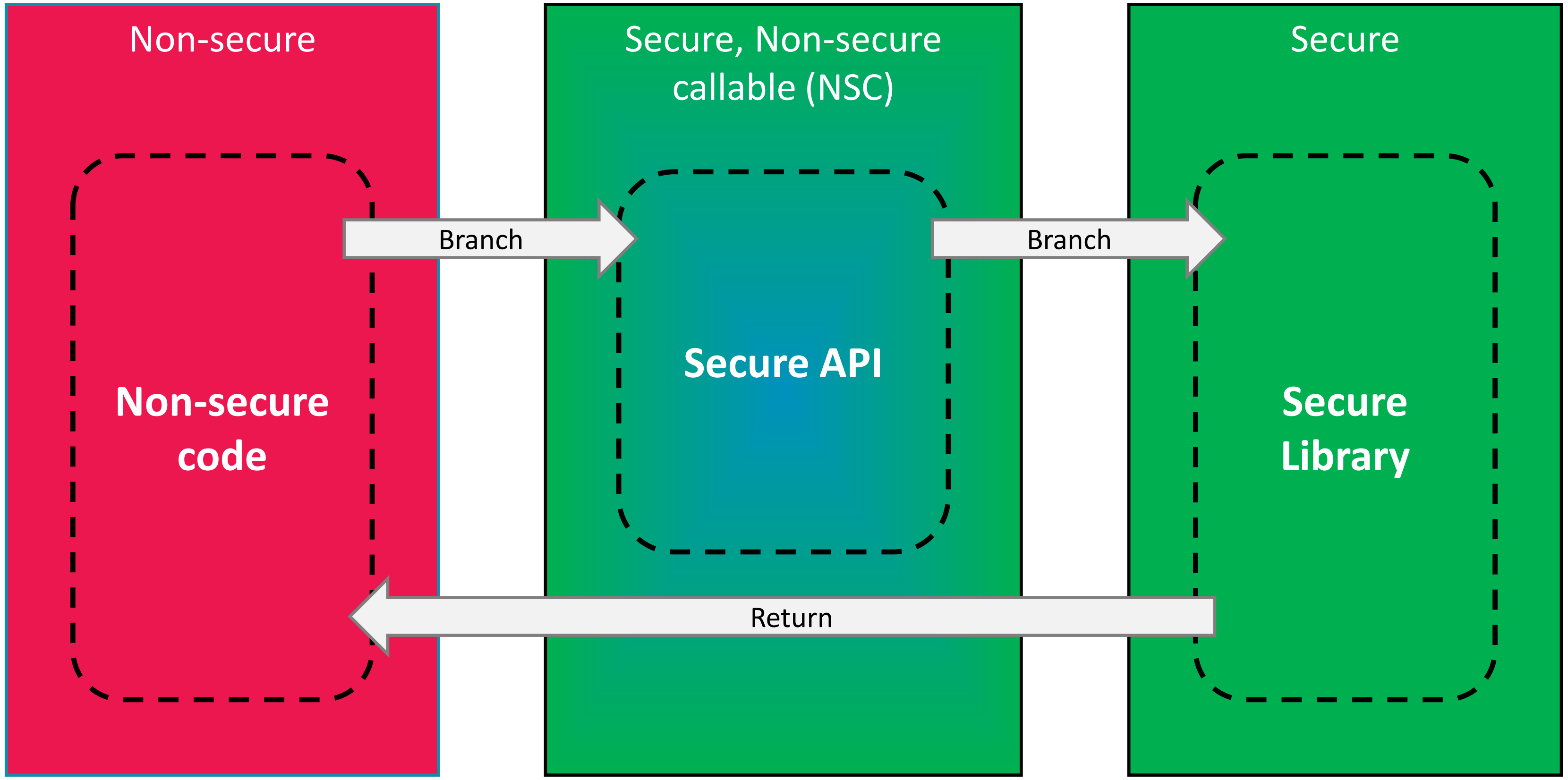
Secure and Non-secure states



Memory security

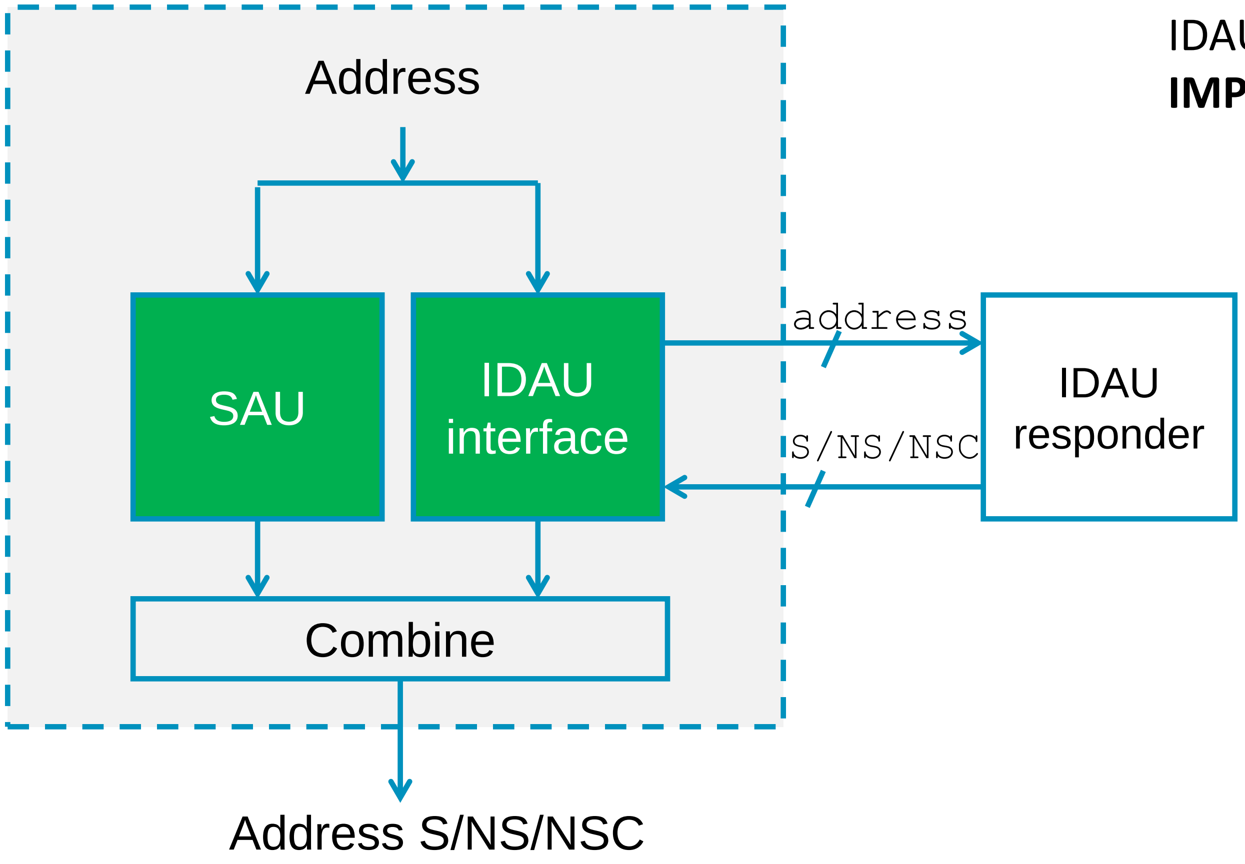
Physical memory is split into Secure and Non-secure regions

- A Secure region can also be Non-Secure Callable (NSC)



Memory security determination

- The security state of a memory region is controlled by the combination of two values
  - Security Attribution Unit (SAU)
  - Implementation Defined Attribution Unit (IDAU)



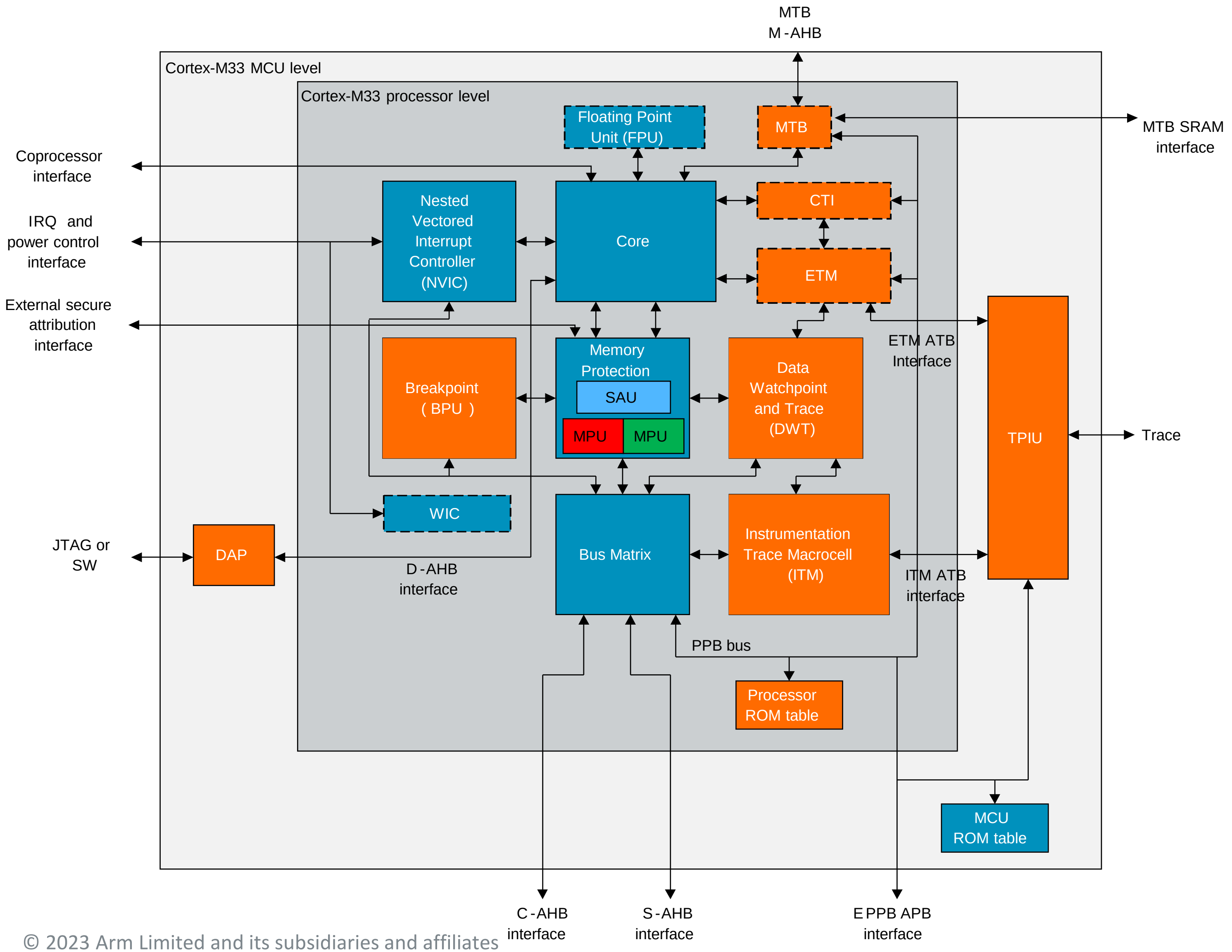
IDAU is optional and its details are **IMPLEMENTATION DEFINED**

Arm Cortex-M processor portfolio

	Armv6-M	Armv7-M	Armv8-M (TrustZone)
High performance		Cortex-M7: Outstanding performance, control and DSP	Cortex-M85: Maximum performance for control, DSP and ML
Performance efficiency		Cortex-M3: Performance efficiency; Cortex-M4: Mainstream control and DSP	Cortex-M33: Flexibility, control and DSP; Cortex-M35P: Tamper resistance, flexibility, control and DSP; Cortex-M55: Balanced performance and efficiency for control, DSP & ML
Lowest power & area	Cortex-M0: Lowest cost, low power; Cortex-M0+: Highest energy efficiency		Cortex-M23: Smallest area, lowest power

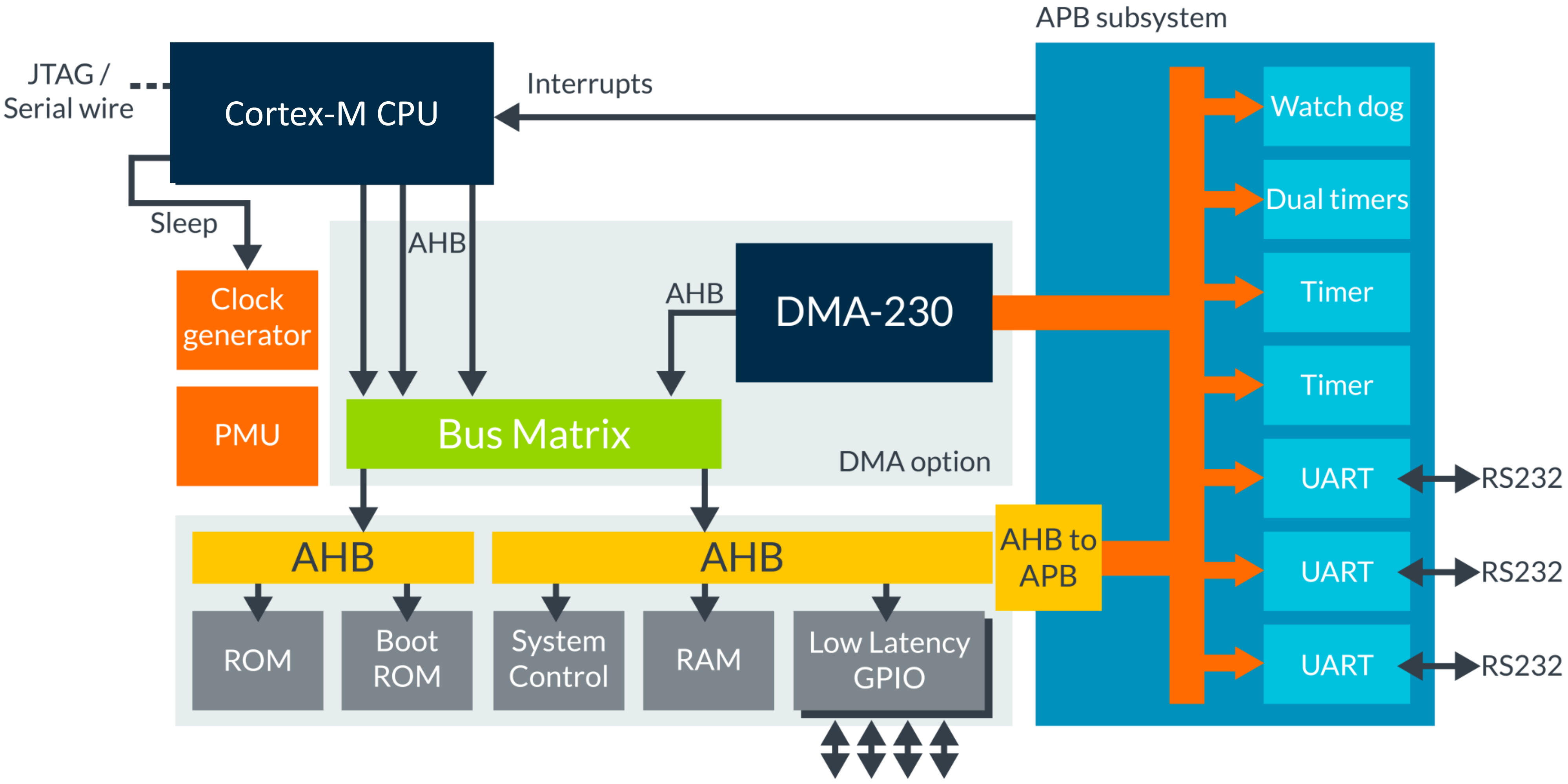
arm

Arm Cortex-M33 Processor



- Mainline profile of Armv8-M
- Synthesizable - configurable RTL
- In-order 2/3 stage pipeline
- Optional Security Extension
- Optional Floating-point Extension (FPv5)
- Optional DSP Extension
- PMSAv8 memory system architecture
- MPU supporting up to 16 regions
- SAU supporting up to 8 regions
- NVIC supporting up to 480 interrupts
- AMBA5 AHB Master Interface
- CoreSight-compliant Debug & Trace
- Ultra-low Power Support with separate power domains
- External coprocessor support
- Optional Custom Datapath Extension(CDE) from r1p0

Example of an Arm based MCU Architecture



Source: https://developer.arm.com/ip-products/subsystem/corstone-foundation-ip/cortex-m-system-design-kit

Summary: Building Embedded Systems using MCUs

- In most embedded systems, MCUs are the best solution as they offer:
  - Low development and manufacturing cost
  - Easy porting and updating; Light footprint
  - Relatively low power consumption
- Increased ‘intelligence’ in embedded microcontroller devices
- An ecosystem of more than 180 Billion Arm-based chips
- Security as a primary design concern
- → Cortex-M provides a strong basis to teach and explore these important concepts

Useful resources

- Free training course on Armv8-M and Cortex-M processors:
  https://www.coursera.org/specializations/cortex-m-architecture-and-software-development

- Architecture Reference Manual:
  https://developer.arm.com/documentation/ddi0553/bp/

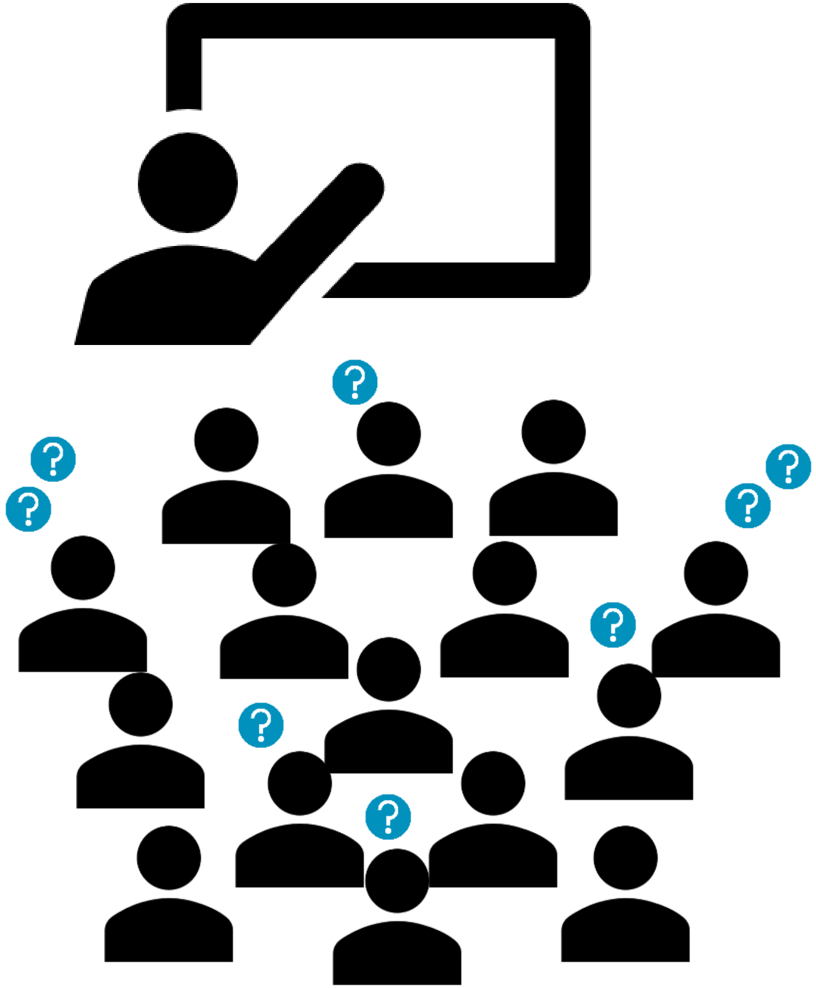
- Cortex-M33 Technical Reference Manual:
  https://developer.arm.com/documentation/100230/latest

- Cortex-M33 Devices Generic User Guide:
  https://developer.arm.com/documentation/100235/latest

- CMSIS:
  https://github.com/ARM-software/CMSIS_5

- Trusted-Firmware-M
  https://www.trustedfirmware.org/projects/tf-m/

Questions & Answers Session



Thank You
Danke
Gracias
Grazie
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה