Embedded System Technologies for Deep Learning and Approximate Computing

Under the SPARC Project P:271 – "Approximate Computing Techniques for Resource Constrained Edge Devices"

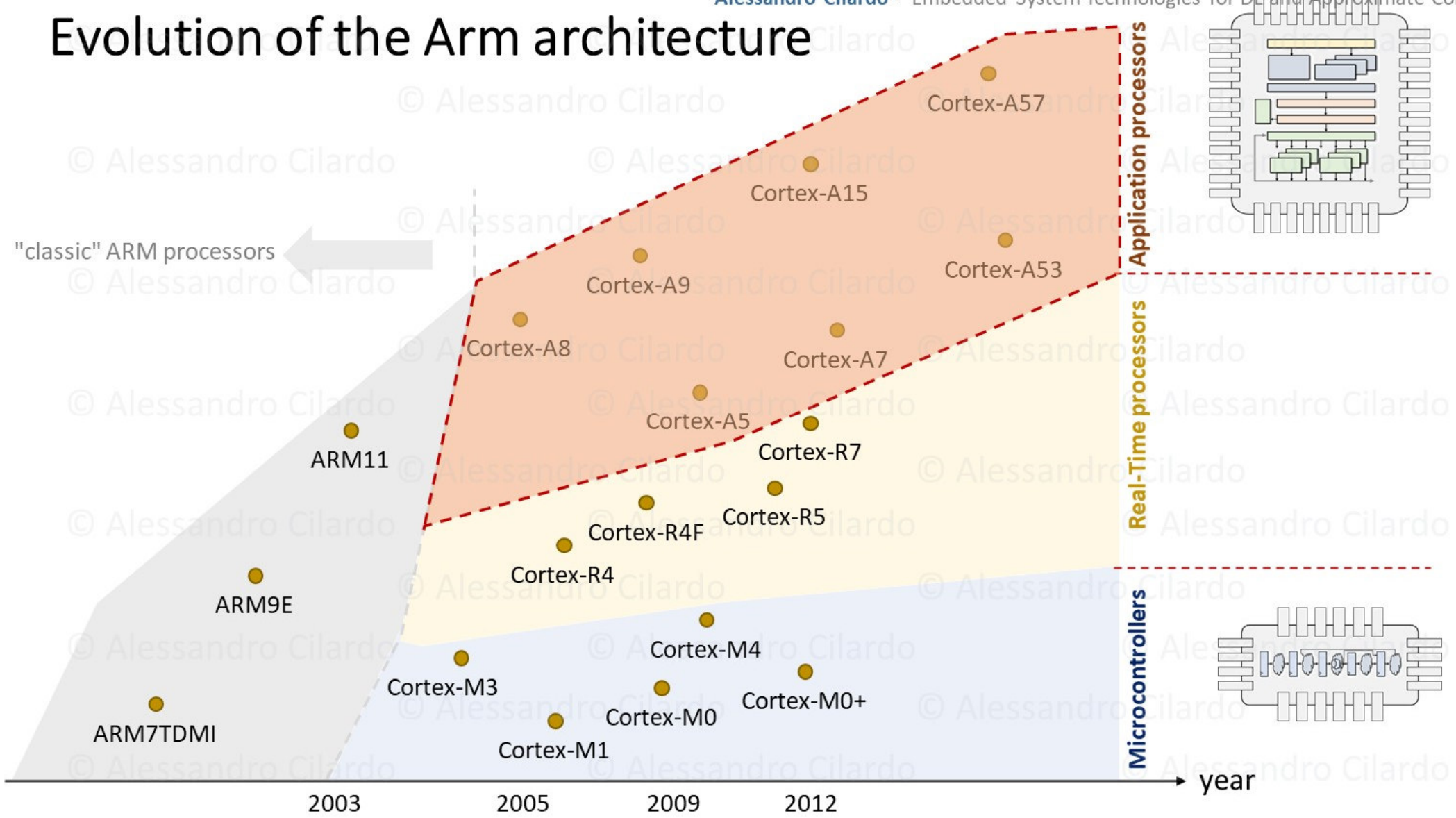
Prof. Alessandro Cilardo

acilardo@unina.it

Day 3
June 5th, 2021

Application-class processors and Multi-Processor Systems-on-Chip

Evolution of the Arm architecture



ARMv7-A cores

- **A**pplication profile (ARMv7-A)
  - memory management support (MMU)
  - High performance at relatively low power
  - driven by multi-tasking OS system requirements
  - e.g. Cortex-A5, Cortex-A8, Cortex-A9, Cortex-A15

Arm Cortex-A: Data Sizes and Instruction Sets

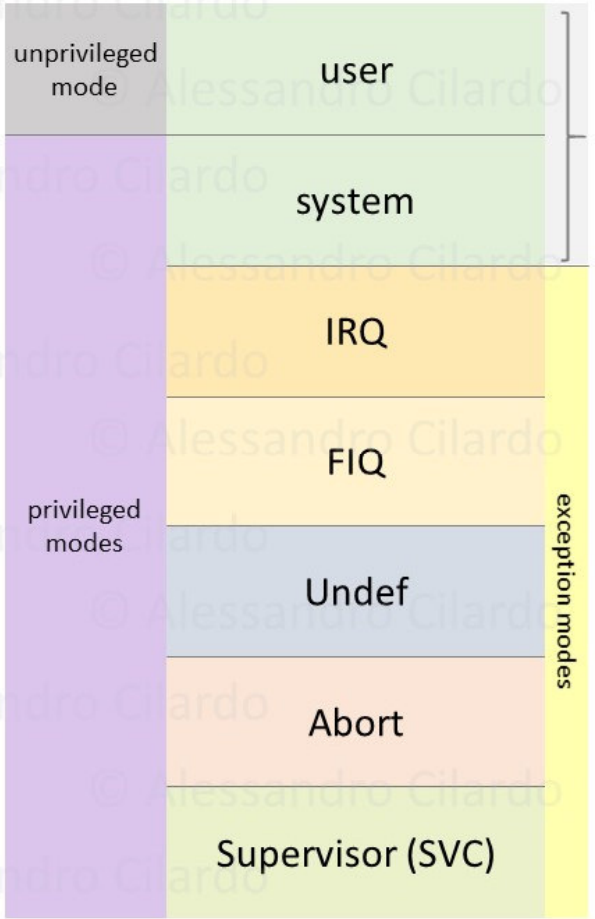
- ARM is a 32-bit load/store RISC architecture
  - The only memory accesses allowed are loads and stores
  - Most internal registers are 32 bits wide
  - Most instructions execute in a single cycle
  - Native data types: Halfword (16 bits), Word (32 bits), Doubleword (64 bits)
- ARM cores implement two basic instruction sets
  - **ARM instruction set**: instructions are all 32 bits long
  - **Thumb-2** instruction set: instructions are a mix of 16 and 32 bits
- Depending on the core, may also implement other instruction sets, e.g.:
  - **VFP** instruction set: 32 bit (vector) floating point instructions
  - **NEON** instruction set: 32 bit SIMD instructions
  - **Jazelle-DBX**: provides acceleration for Java VMs (with additional software support)
  - **Jazelle-RCT**: provides support for interpreted languages

Arm Cortex-A: Data Sizes and Instruction Sets

- Data Processing Instructions
  - do not affect memory, only work on registers
  - **`ADD`**, **`SUB`**, **`AND`**, **`XOR`**, **`CMP`**, **`CMN`**, **`TST`**, **`TEQ`**, . . .
- Single Access Data Transfer
  - Use to move data between one or two registers and memory
  - **`LDR`**, **`LDRD`**, **`STR`**, **`STRB`**, **`STRH`**,. . .
- Multiple Register Data Transfer: **`LDM`**, **`STM`**
  - These instructions move data between multiple registers and memory
  - (*note, single/multiple push/pop operations are equivalent to single/multiple LD/ST*)
- Subroutines
  - supported by a branch-&-link instruction: **`BL`**
  - return address stored in the link register (**`r14`**, aka **`lr`**), return just branches to the address in **`lr`**
  - range of the branch target address depends on instruction set and width
- Special instructions, e.g. Supervisor Call (**`SVC`**)
  - causes an SVC exception, then SVC handler examines the SVC number determining the requested operation
  - OS can implement a set of privileged operations (*system calls*) which user mode applications can request

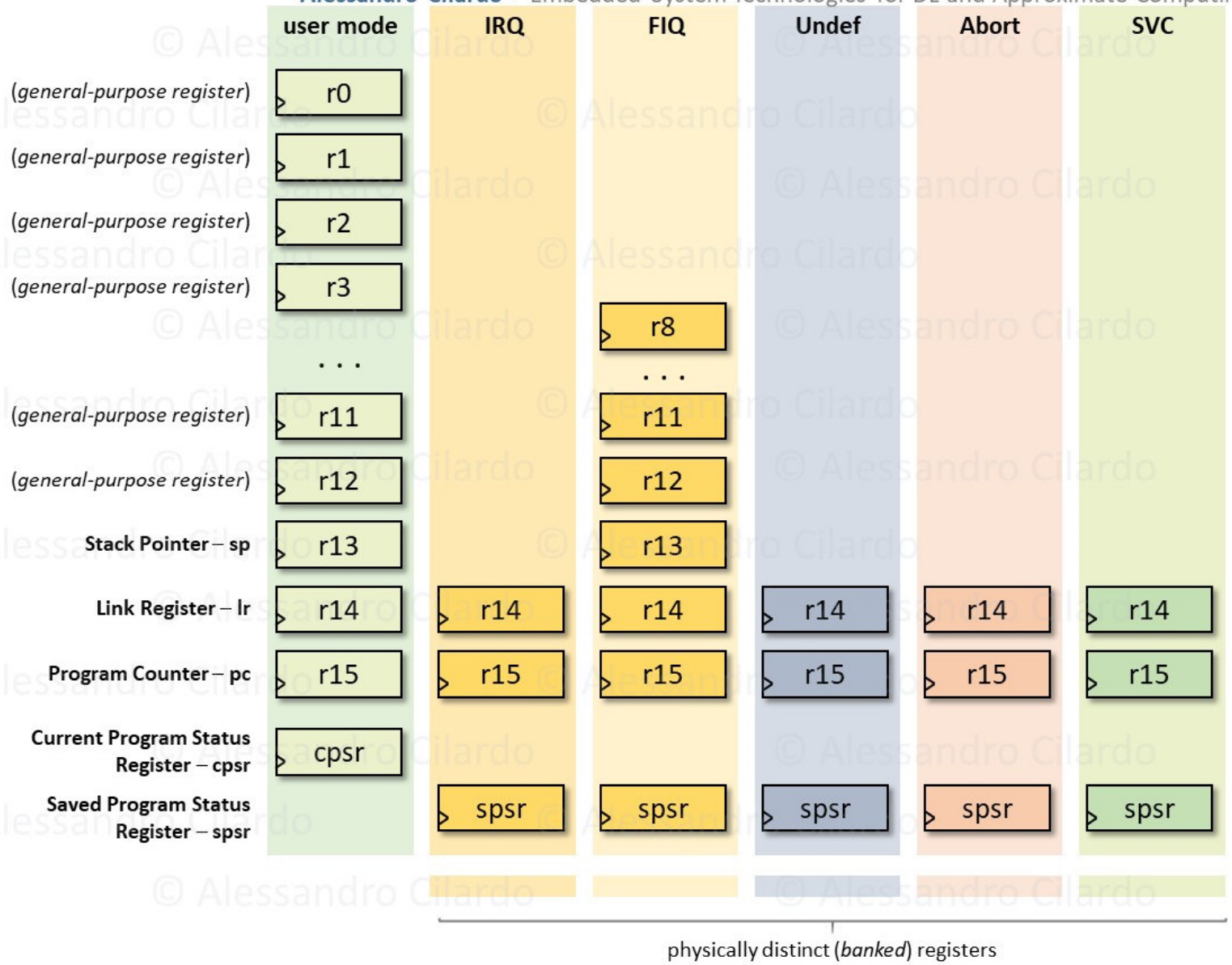
Processor modes

- Seven basic operating modes
  - each mode accesses its own stack space and a subset of registers
  - some operations can only be carried out in a *privileged* mode
- Supervisor (SVC)
  - entered when a SVC instruction is executed
- FIQ, IRQ
  - resp, high-priority and normal priority interrupt modes
- Abort
  - entered when a memory access violation is raised
- Undef
  - handles undefined instructions
- System
  - privileged mode seeing the same registers as User Mode
- User
  - unprivileged mode for normal code

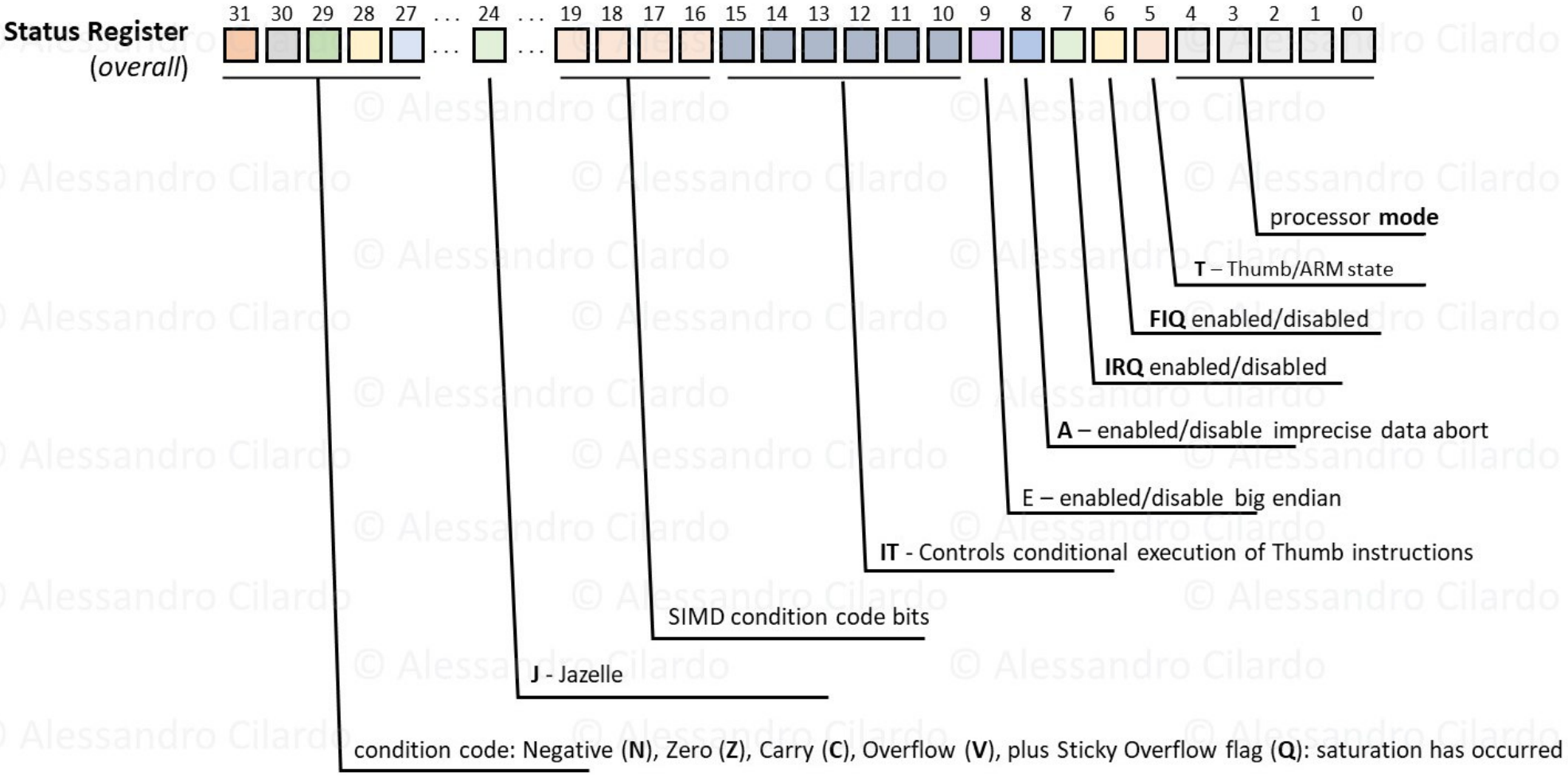


Register file

- a *banking* mechanism is used for fast mode switch
  - e.g. the status register (**`cpsr`**) is automatically copied into the "saved" program status register (**`spsr`**) of the entered mode

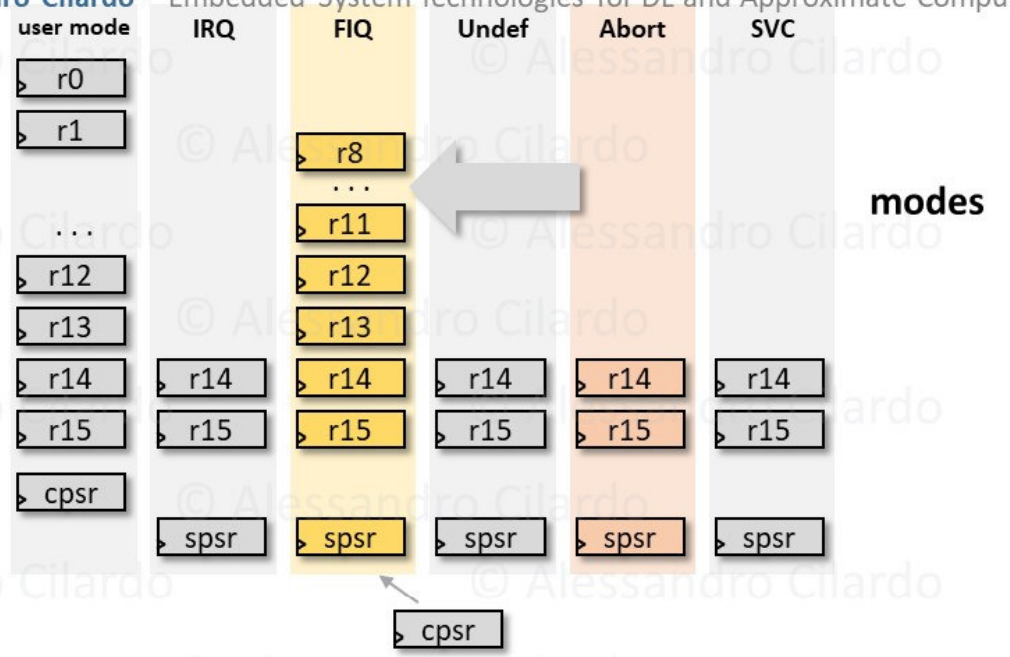


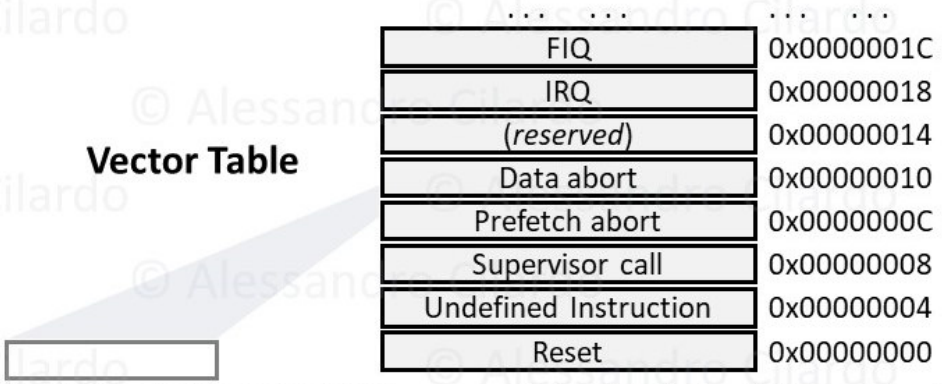
Status register



Exception handling

- When an exception occurs, the core
  - copies **CPSR** into ***SPSR_mode***
  - sets appropriate **CPSR** bits
    - change to ARM state (if appropriate)
    - change to exception mode
    - disable interrupts (if appropriate)
  - stores the return address in ***LR_mode***
  - sets **PC** to vector address
- To return, exception handler needs to…
  - restore **CPSR** from ***SPSR_mode***
  - restore **PC** from ***LR_mode***
- On an exception, cores can enter ARM state or Thumb state
  - controlled through settings in **CP15**
- Very different from the **v7-M / v6-M** exception model



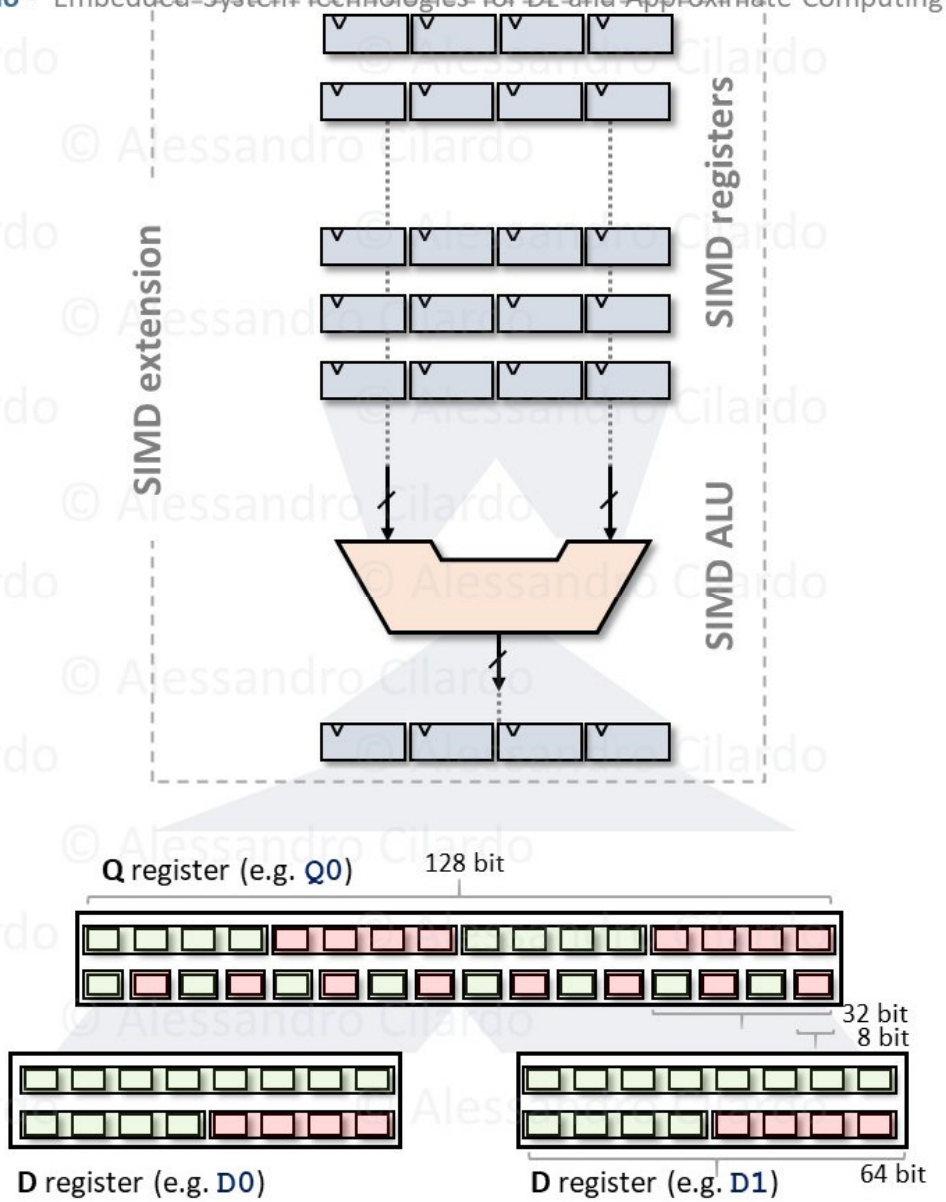


a branch instruction to the appropriate Handler is stored in each Vector Table entry

Note: The Vector Table can also be placed at address 0xFFFF0000.

ARM Neon extensions

- ARM architectural support for SIMD-like operations (instruction set extensions)
  - ARMv6: introduced support for SIMD instruction (~2002)
  - ARMv7: introduced Neon SIMD extensions (~2009)
  - earlier versions (≤v5) supported Vector Floating Point (VFP) extensions, offering more basic functions
- ARMv7 **Advanced SIMD**, also known as **Neon**
  - 16 additional *128-bit* registers (**Q0**-**Q15**), which can also be viewed as 32 *64-bit* registers (**D0**-**D31**)
  - Data types: signed/unsigned 8-bit, 16-bit, 32-bit, 64-bit integral types, single precision floating point
  - Additional QC integer saturation summary flag (sticky) in the FP status and control register (FPSCR)

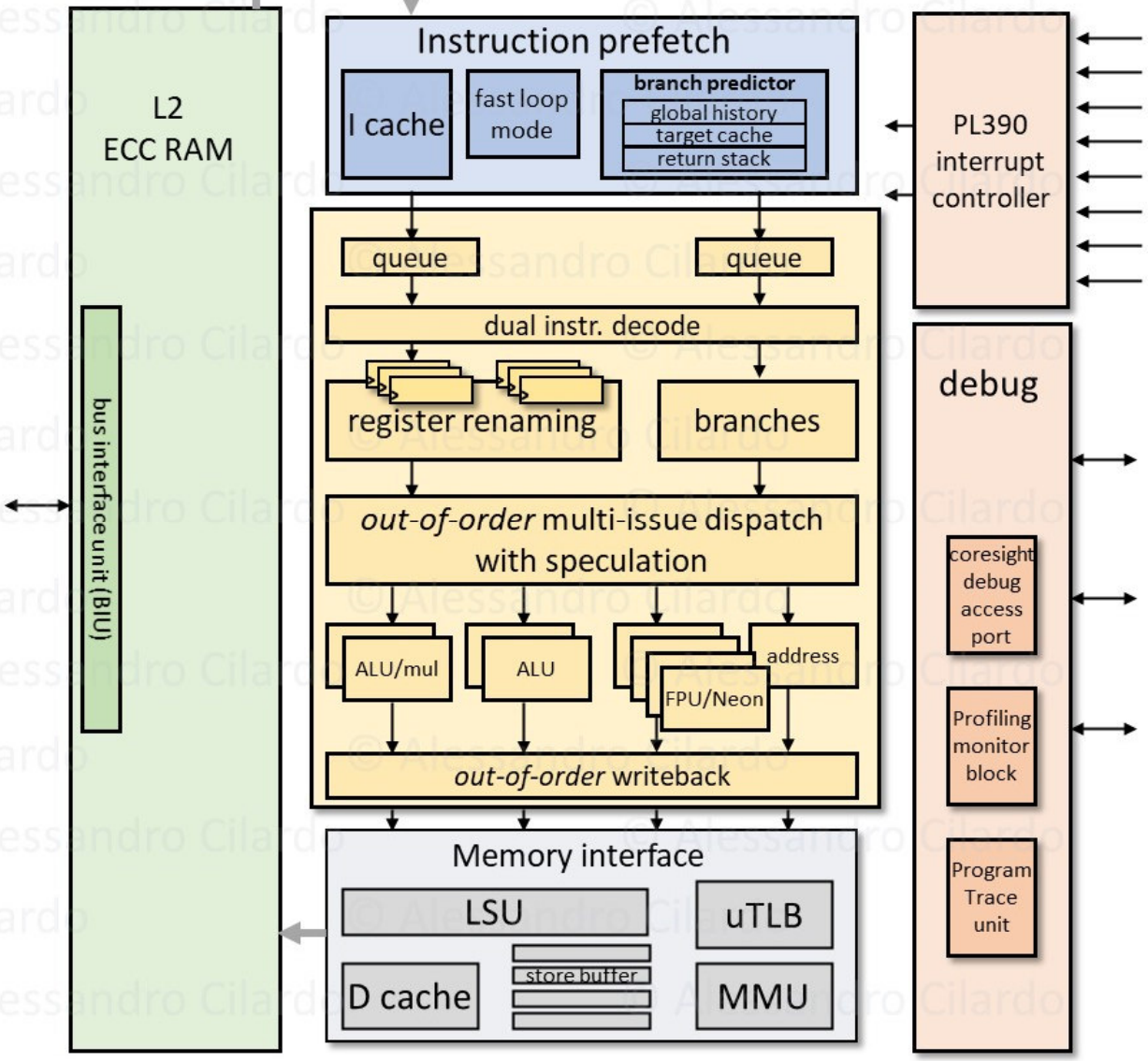


Cortex-A memory model

- Each defined memory region will specify a memory type
- The memory type controls:
  - Memory access ordering rules
  - Caching and buffering behaviour
- Three, mutually exclusive memory types:
  - Normal
  - Device
  - Strongly Ordered
- Normal and Device memory allow additional attributes for specifying
  - the cache policy
  - whether the region is Shared
  - normal memory allows you to separately configure Inner and Outer cache policies

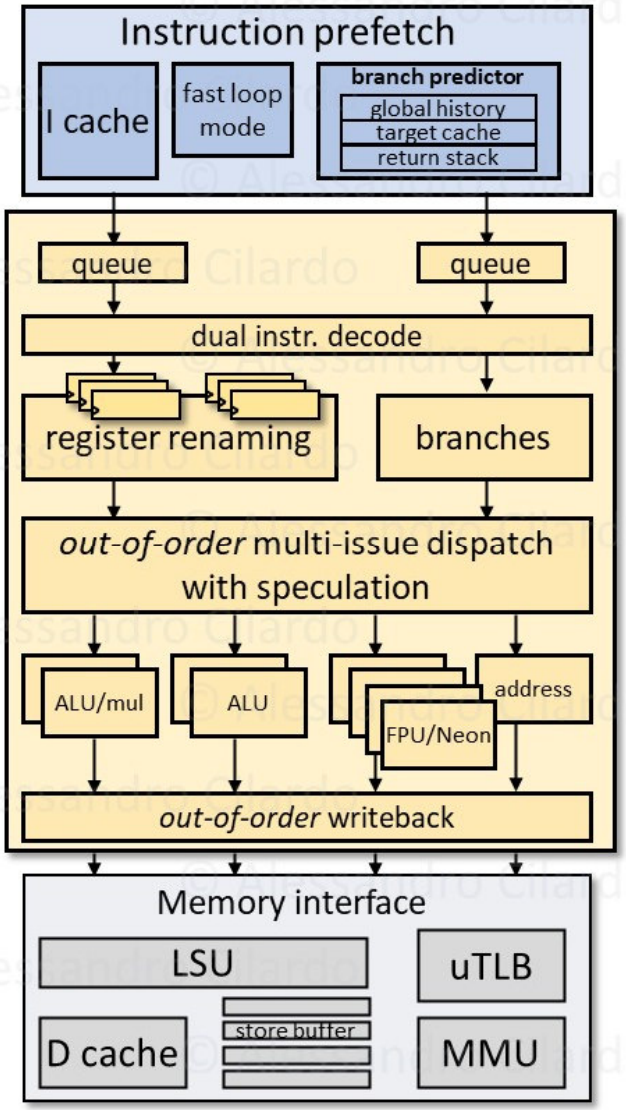
Case study: Cortex A9

- ARMv7-A Architecture
  - Thumb-2, Thumb-2EE
  - TrustZone support
- Variable-length multi-issue pipeline
  - Register renaming
  - Speculative data prefetching
  - Branch Prediction & Return Stack
- 64-bit AXI instruction and data interfaces
- TrustZone extensions
- L1 Data and Instruction caches
  - 16-64KB each
  - 4-way set-associative
- Optional features:
  - PTM instruction trace interface
  - IEM power saving support
  - Full Jazelle DBX support
  - VFPv3-D16 Floating-Point Unit (FPU) or NEON™ media processing engine



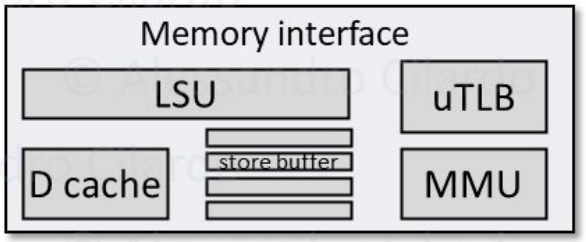
Case study: Cortex A9

- Instruction cache size: 16KB, 32KB, or 64KB
- Superscalar pipeline: fetching two instructions at once
- Branch Prediction:
  - Global History Buffer: 1K ~ 16K entries
  - Branch-Target Address Cache: 512 ~ 4K entries
  - Return stack of 4 x 32 bits
- Fast-loop mode
  - instruction loops < 64 bytes can be buffered with no I-cache accesses
- Superscalar Decoder
  - can decode two full instructions per cycle
- Register Renaming
  - resolve data dependencies and unroll small loops by hardware
- Issue can be fed with a maximum of two instructions per cycle
- Issue can dispatch up to four instructions per cycle
- Out-of-order selection of instructions from queue
- Variable length Executing Stage (1 - 3 cycles)
  - most Instructions finish within one cycle
  - instruction which folds shifts and rotates can take three cycles
    - ADD r0, r1, r2 (1 cycle)
    - ADD r0, r1, r2 LSL #2 (2 cycle) → Corresponds to a = b + (c << 2);
    - ADD r0, r1, r2 LSL r3 (3 cycle) → Corresponds to a = b + (c << d);



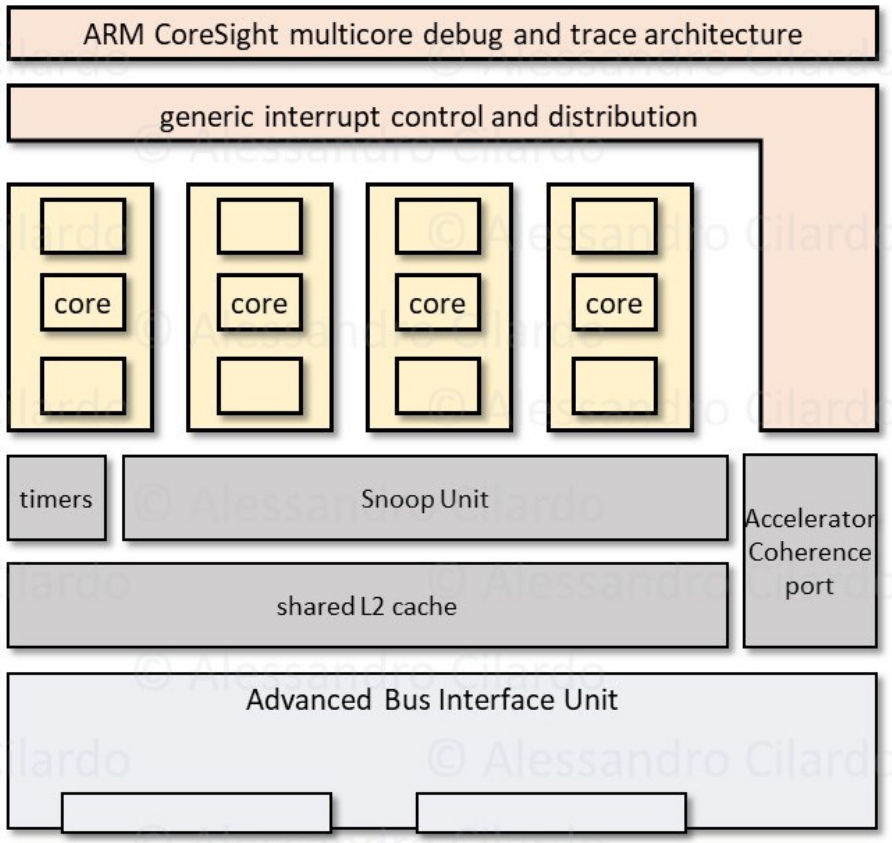
Case study: Cortex A9

- Two-level TLB structure
  - micro TLB
    - 32 entries on data side and 32 or 64 entries on instruction side
    - to reduce power consumed in translation and protection look-ups
  - main TLB
- Data prefetcher
  - monitor cache line requests and cache misses to control data prefetching
  - can prefetch up to eight independent data streams
  - prefetch and allocate data in the L1 data cache, as long as it keeps hitting in the prefetched cache line
  - dependent load-store instructions forwarded for resolution within memory system



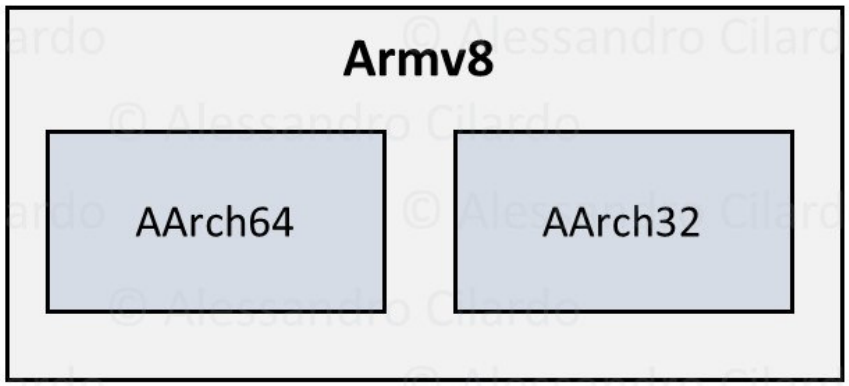
Cortex-A MPCore processors

- Standard Cortex cores, with additional logic to support **multi-processor** architectures
  - called *MPCore* variants
  - 1 to 4 CPUs
- Includes integrated
  - Interrupt controller
  - Snoop Control Unit (SCU)
  - Timers and Watchdogs



ARMv8-A

- **AArch32**: A32+T32 ISAs
  - scalar Floating-Point support (SP and DP)
  - advanced SIMD (SP float)
- **AArch64**: A64 ISA
  - scalar FP (SP and DP)
  - advanced SIMD (SP+DP float)
- 64-bit pointer and registers
- Fixed length (32-bit) instructions
- Load/store architecture
- Little endian (but can be configured as big endian)
- 31 general-purpose registers and zero register
- Unaligned accesses (except for exclusive and ordered accesses)
- ARMv7-A compatibility: Thumb2, TrustZone, VFPv3/v4, NEON, Adv SIMD,..



ARMv8-A: evolution from v7

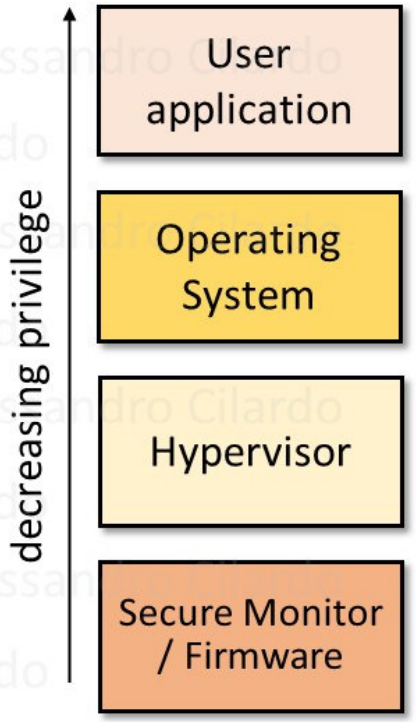
- v7 features that were dropped:
  - conditional execution of most instructions
    - e.g. removed the complex T32 IT*xxxx* instruction handling *if-then-else* patterns
  - shifts combined with arithmetic instructions
    - immediate shifts only
    - no RRX shift, no ROR shift for ADD/SUB
  - "load multiple" (LDM) and "store multiple" (STM) used for push and pop operations
  - read/write access to PC register
  - co-processor integration
    - ARMv8-A relies on system instructions
- Previous features that were kept
  - VFP and floating point support
    - FP support mandatory in ARMv8-A
  - AdvSIMD based on NEON but with major changes
  - weakly ordered memory
  - similar basic arithmetic instructions
- Added features
  - load-acquire and store-release atomics
  - cryptographic instructions: AES and SHA
  - AdvSIMD available for general-purpose floating point operations
  - larger PC-relative branching
    - Literal pool access and most conditional branches are extended to ± 1MB
    - unconditional branches and calls to ±128MB
  - non-temporal (cache skipping) load/store
  - load/store of a non-contiguous pair of registers

ARMv8-A: registers and data model

- 64 Bit integer registers:
  - X0 - X29, X30/LR, SP/ZERO
  - the lower 32 bits of each registers are referred to as W0 .. W30
- Register X31 has special behavior depending on the type of access
  - acts as both stack pointer and a zero register:
  - *Zero Register* - it returns 0 when used as a source register, while it discards the result when used as a destination register
  - *Stack Pointer* - it acts as SP when used as a load/store base register and for some arithmetic instructions
- X30/LR for procedure call link register is no longer banked
  - exceptions save PC using the target exception level's **`ELR`** system register

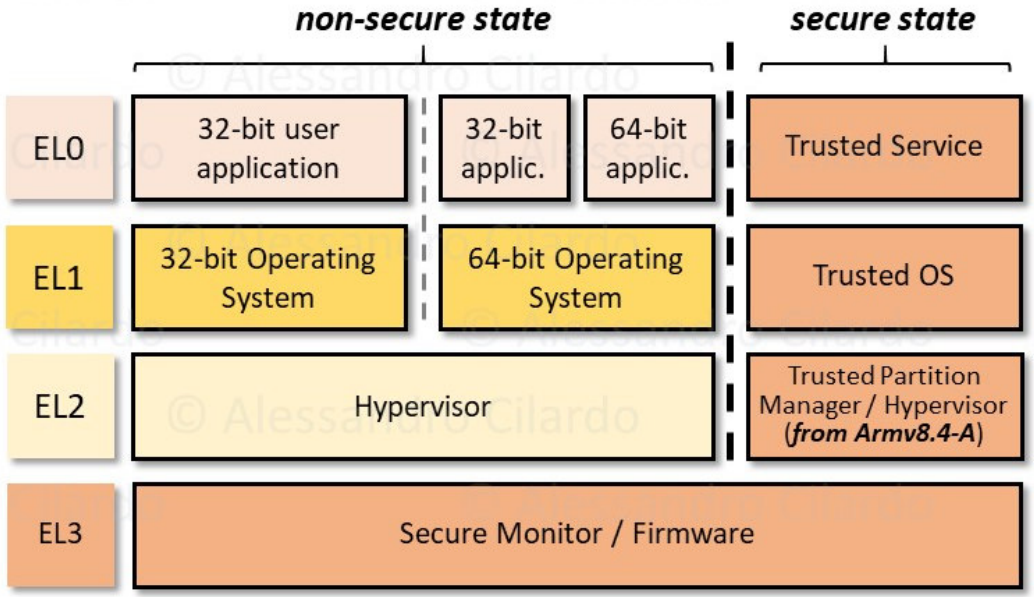
Privileges and Exception levels in Armv8

- Exception Levels (0-3)
  - **EL0**: application code
  - **EL1**: operating system
  - **EL2**: hypervisor
  - **EL3**: low-level firmware and security code
- Two types of privileges
  - **Memory** (attributes assigned to memory regions by MMU)
    e.g. read/write, differentiated for privilege (EL0 vs. EL1..EL3)
  - **System Registers** (define the current processor *Context*)
    Multiple physical registers, differentiated by EL, controlling a number of features (e.g. MMU configuration, address spaces, etc.)
    e.g. the **`TTBR0_EL1`** register holds the base address of the translation table for EL0 and EL1 (it can be accessed from EL1, but not from EL0)



Execution and Security states

- Overall state: (Exception Level + Execution state + Security state)
- **Execution State**: Armv7 **AArch32** (A32+T32) *vs.* Armv8 **AArch64**
  - affects aspects of the memory models and how exceptions are managed
- **Security state**: **Secure** *vs.* **Non-Secure**, based on Arm TrustZone
  - Secure state: processor can access secure *and* non-secure address spaces, System Registers, and can *only* ack Secure Interrupts
  - Non-Secure state: only non-secure address spaces, System Registers, and Interrupts
- Changing Execution State
  - AArch32→64 changes only allowed when *moving to higher ELs*, and vice versa, so..
  - a 64-bit OS can host 32-/64-bit applications
  - a 32-bit OS can only host 32-bit applications
  - Similar rules apply to hypervisors
- Changing Security State
  - controlled by EL3 (always in Secure state)
  - decides the Security States of lower ELs
  - Security State can only change on EL switch

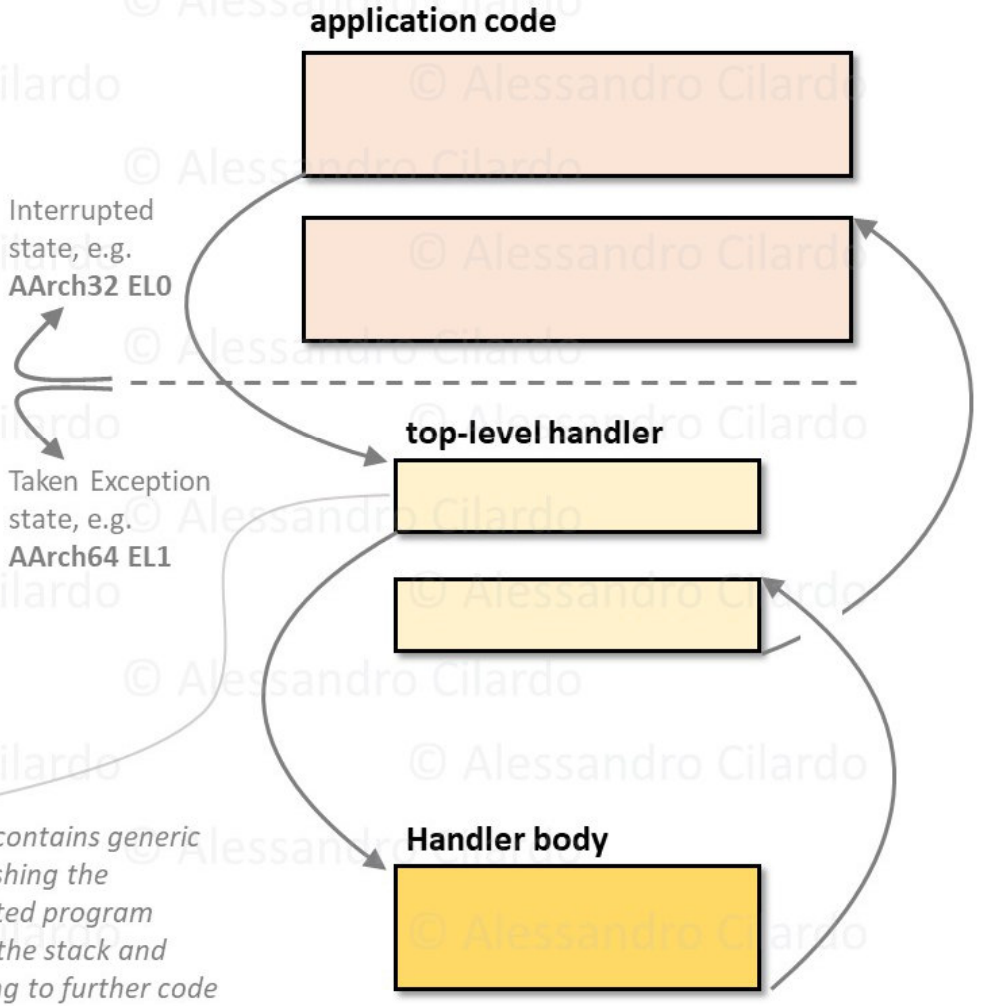


Exception types

- Synchronous exceptions:
  - invalid instructions, memory accesses (misalignment or MMU permissions), debug,..
  - exception-generating instructions: `SVC`, `HVC`, and `SMC`, used to implement system calls allowing less privileged code to request services from more privileged code
- Asynchronous exceptions:
  - can happen at any time, can be masked (leave the request in a pending state)
  - *Physical* interrupts: SError (System Error), IRQ, FIQ
  - *Virtual* Interrupts: vSError (Virtual System Error), vIRQ (Virtual IRQ), vFIQ (Virtual FIQ)
  - IRQ and FIQ types: used for **peripherals**. In Armv7, FIQ has higher priority. In Armv8, the two are use to implement the *Non-Secure* and *Secure* interrupt differentiation
  - SError: External memory abort (bus or ECC error). Considered asynchronous as the memory instruction might have already been retired

Exception handling

- Exception return address is saved in **`ELR_Elx`**, interrupted PSTATE in register **`SPSR_Elx`** (ELx is the *target* level)
  - the two will be atomically restored by the **`ERET`** AArch64 instruction (*note*: General-purpose registers to be preserved by software)
- Asynchronous exceptions routing
  - Physical interrupts (SError, IRQ, FIQ) routed to one of EL1, EL2 or EL3 (as set in system configuration registers)
  - Interrupts routed to lower ELs are masked (i.e. left pending) when executing in higher levels
  - The new execution state may be Secure or Non-Secure, as configured in higher ELs
- Exception stacks in AArch64: EL0 stack (always accessible in the new EL) along with the target EL stack
- Vector tables held in normal memory, set by software. Base address **`VBAR_ELn`** set by a configuration register.
  - Each of EL1..3 has its own vector table. Each type can branch to one of 4 locations based on the state of the interrupted EL



Exception handling: vector tables

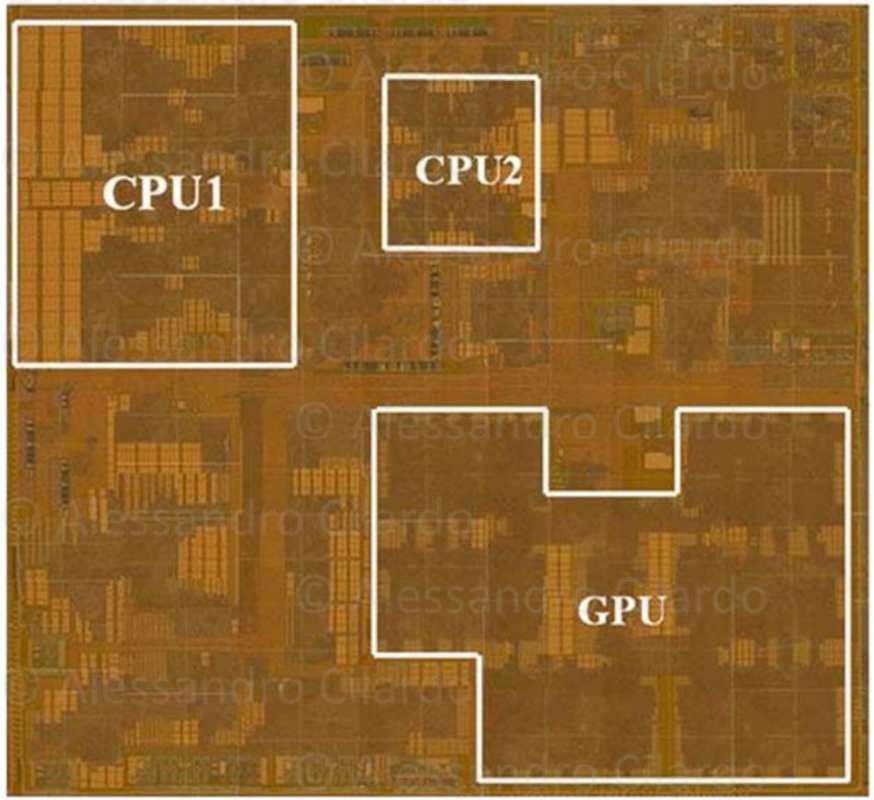
VBAR_ELn + 0x780	Serror / vSError	*Exception from a lower EL and all lower ELs are AArch32*
... ... + 0x700	FIQ / vFIQ	
... ... + 0x680	IRQ / vIRQ	
... ... + 0x600	Synchronous	
... ... + 0x580	Serror / vSError	*Exception from a lower EL and at least one lower EL is AArch64*
... ... + 0x500	FIQ / vFIQ	
... ... + 0x480	IRQ / vIRQ	
... ... + 0x400	Synchronous	
... ... + 0x380	Serror / vSError	*Exception from the current EL while using* SP_ELx
... ... + 0x300	FIQ / vFIQ	
... ... + 0x280	IRQ / vIRQ	
... ... + 0x200	Synchronous	
... ... + 0x180	Serror / vSError	*Exception from the current EL while using* SP_EL0
... ... + 0x100	FIQ / vFIQ	
... ... + 0x080	IRQ / vIRQ	
VBAR_ELn + 0x000	Synchronous	

Case study: ARM Cortex-A53

- Implements the ARMv8 ISA. Introduced in 2012. Main target: low-power systems
  - typically used in SoC for entry-level smartphones or medium-end embedded devices
- Licensed to
  - AMD, Broadcom, Samsung, Altera, Stmicroelectronics, MediaTek, Qualcomm, Xilinx, ..
- Often combined with higher performance cores in ARM **big.LITTLE** configuration
  - e.g. Cortex-A53 + Cortex-A57 or Cortex-A72
  - provides a configurable mix of compute resources
  - allows various performance/power trade-offs
- Microarchitecture aspects:
  - 8-stage pipelined processor with 2-way superscalar, in-order execution pipeline
  - DSP and NEON SIMD extensions are mandatory per core
  - VFPv4 Floating Point Unit onboard (per core)
  - Hardware virtualization support, TrustZone security extensions
  - 64-byte cache lines, 10-entry L1 TLB, and 512-entry L2 TLB
  - 4 KiB conditional branch predictor, 256-entry indirect branch predictor

Example: Samsung Exynos 5433

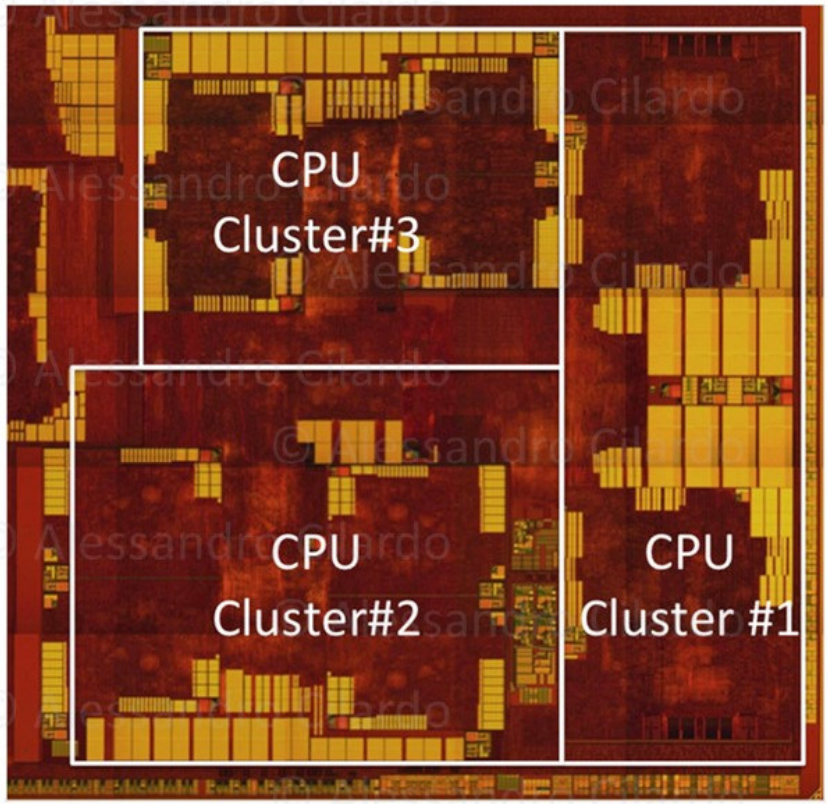
- Samsung 20nm process
- 113 mm² die size
- Mali-T760 (6 EU)
- Quad-core Cortex-A53 (small cores)
  - 32 KiB L1I$ and 32 KiB L1D$ per core, and a shared 256 KiB L2
  - 4.4 mm² per cluster (~1 mm² per core, ~0.55 mm² for 256 KiB L2 cache)
- Quad-core Cortex-A57 (big cores)
  - 48KB L1I$ and 32KB L1D$ per core, and a shared 2 MiB L2
  - 15.85 mm² per cluster
  - ~3 mm² per core, ~3.87 mm² for 2 MiB L2 cache



source: https://www.wikichip.org

Example: MediaTek Helio X20

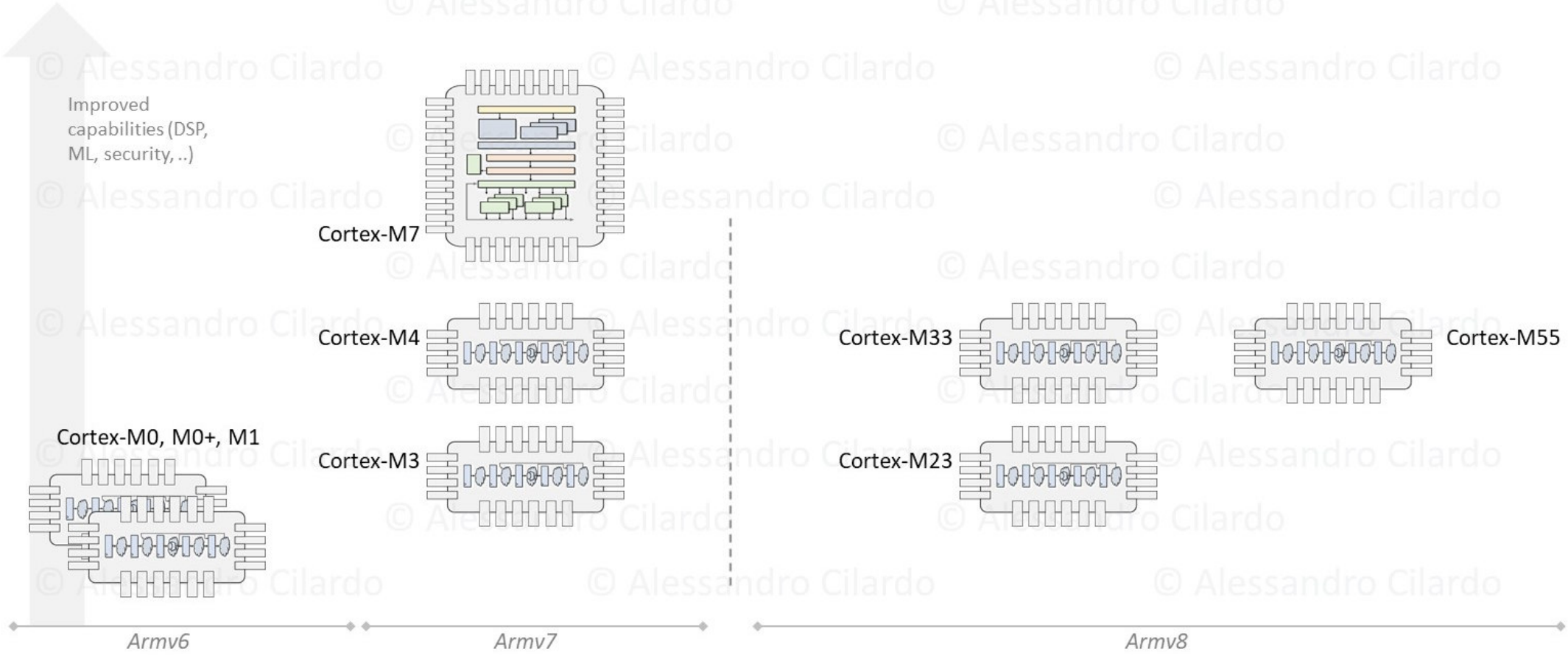
- TSMC 20 nm process
- 100 mm² die size
- Quad-core ULP Cortex-A53
  - ~21.81 mm² per cluster (~4.23 mm² per core)
- Quad-core efficient Cortex-A53
  - ~29.73 mm² per cluster (~5.41 mm² per core)
- Dual-core High-performance Cortex-A72 + 1 MiB L2
  - ~27.36 mm² per cluster (~ 9.60 mm² per core, ~ 7.50 mm² for 1 MiB L2)



source: https://www.wikichip.org

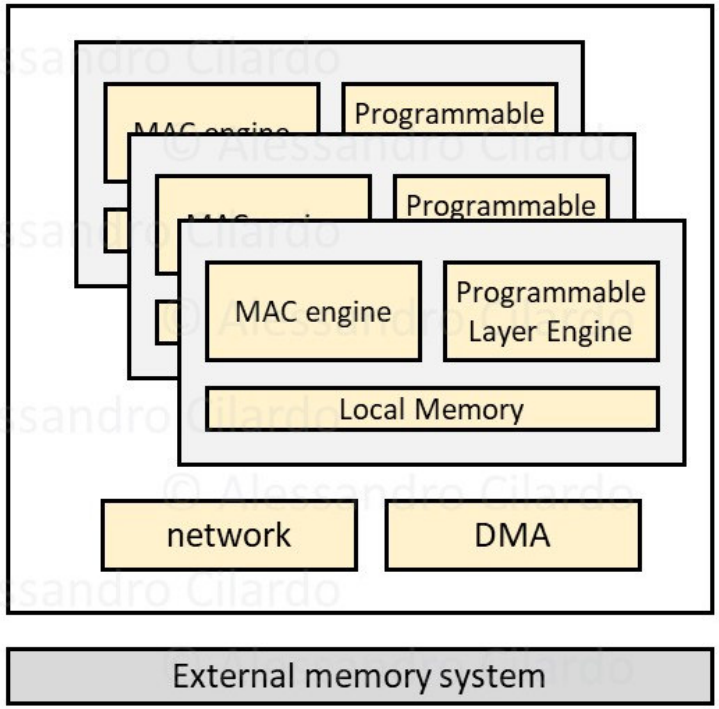
Caveat: back to the M class

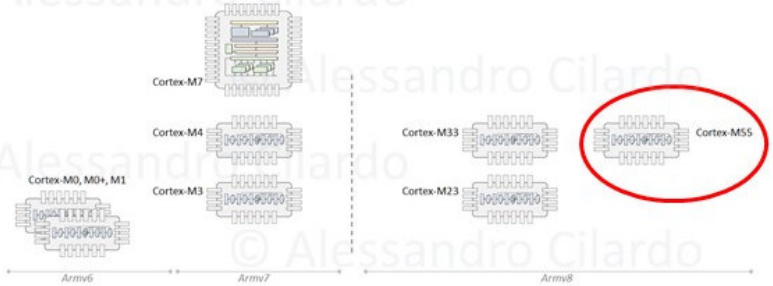
- v8 architecture also covers the M class, since Cortex-M23 (2016)



Arm Ethos Network Processing Unit (NPU)

- Ethos-N processors, Ethos-U processors
- high-performance AI
  - e.g. for Full HD, real-time object detection, super resolution and segmentation
- Performance around 1 TOP/s - +650 TOP/s
- Multicore support (up to eight NPUs)
- A further layered security to protect models and input data, e.g. biometrics for financial payments
- ACE-Lite master port and optional SMMU integration
- Allows early performance feedback by using the NPU static performance analyzer
- coupled with a **Cortex-M55**, can achieve a performance improvement as high as 480X
  - for matrix multiplication, with int8 data representation
  - (preliminary results from ARM, 2020)





FPGA technologies: an overview

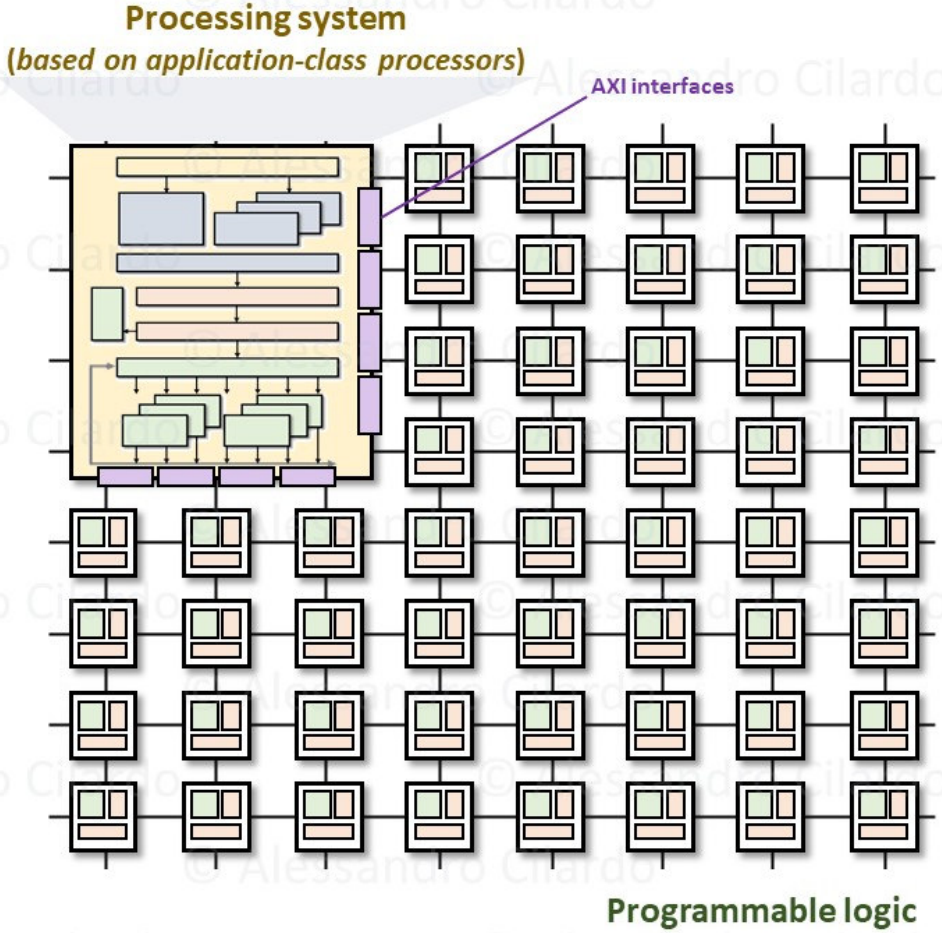
Arm Cortex-A9 based Xilinx Zynq 7000 System-on-Chip

- Essential insight: SoC + HW
- Dual-core ARM Cortex-A9 processor
- medium-end processor capable of running full operating systems such as Linux

+

- Traditional Field Programmable Gate Array
- Xilinx 7-series FPGA architecture

  (28K-235K logic cells)
- Systems on Programmable Chip
  - Zynq-7000: Xilinx “All-Programmable SoC"
- ARM Cortex-A9 cores can coexist with custom accelerators (including soft-core processors)



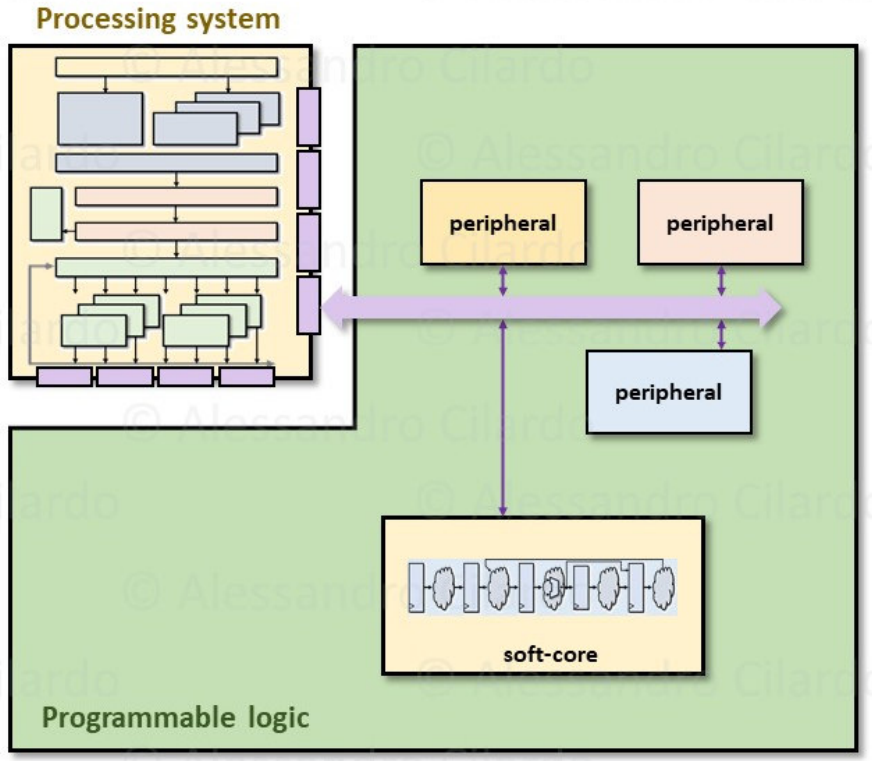
Arm Cortex-A9 based Xilinx Zynq 7000 System-on-Chip

- Essential insight: SoC + HW
- Dual-core ARM Cortex-A9 processor
- medium-end processor capable of running full operating systems such as Linux

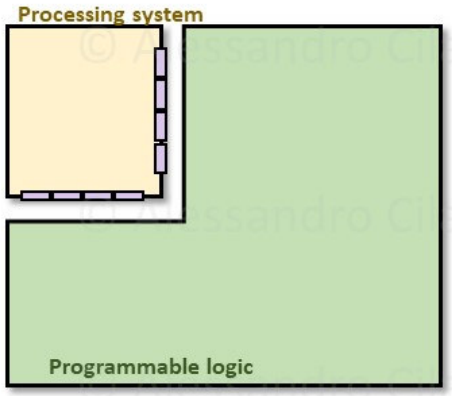
+

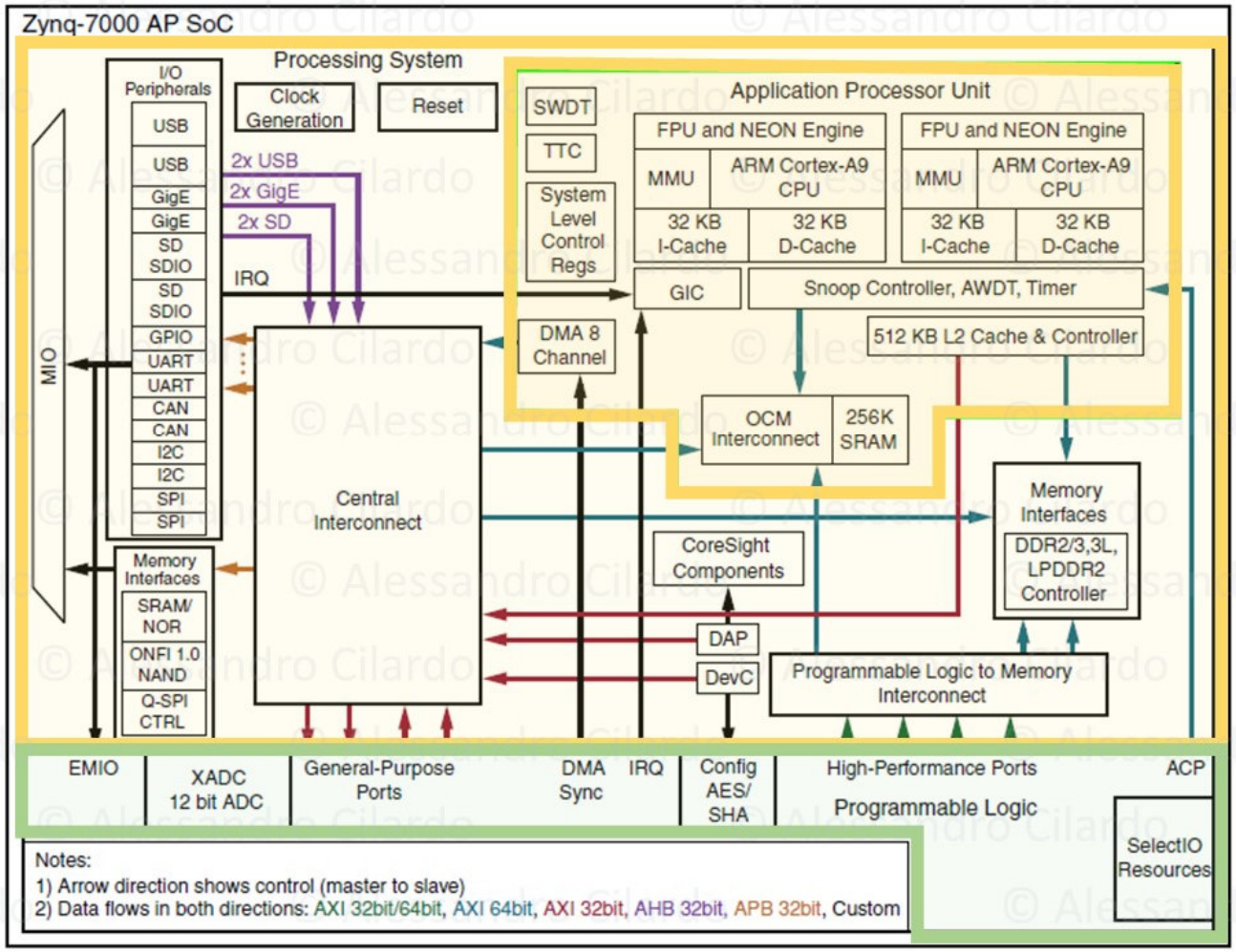
- Traditional Field Programmable Gate Array
- Xilinx 7-series FPGA architecture

  (28K-235K logic cells)
- Systems on Programmable Chip
  - Zynq-7000: Xilinx “All-Programmable SoC"
- ARM Cortex-A9 cores can coexist with custom accelerators (including soft-core processors)



Xilinx Zynq 7000 System-on-Chip: detail





PS Frequency: up to 1GHz

PL Frequency up to 741MHz

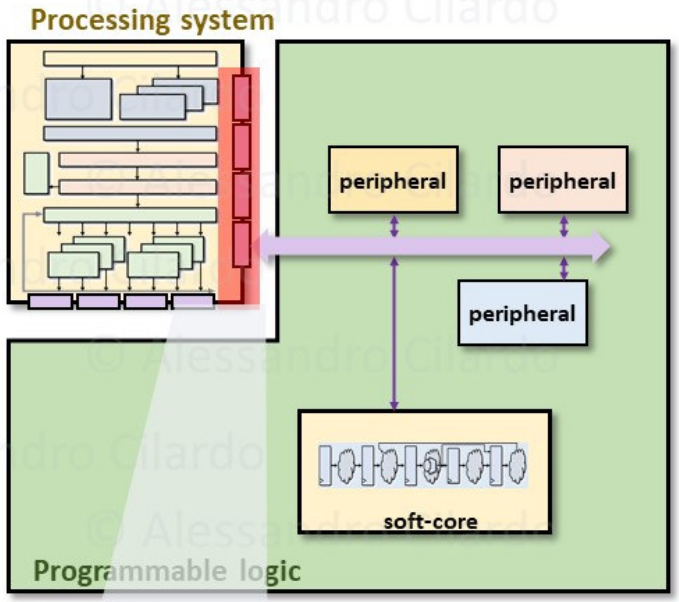
External interfaces and Peripherals

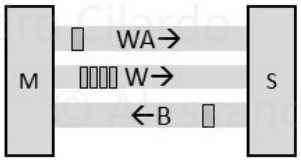
- General Purpose Input/Output (IOBs)
  - High Performance (HP) or High Range (HR)
- Communication Interfaces
  - GTX Transceivers supporting many standard interfaces: PCI Express, Serial RapidIO, SCSI, SATA
  - rates of up to 12.5Gbps are supported
- Analogue to Digital Conversion
- Clocks
- Programming and Debug
  - JTAG ports in the PL for configuration and debugging

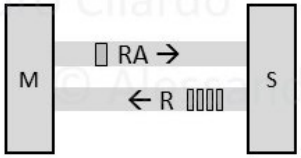
SPI (x2)	**Serial Peripheral Interface** *De facto standard for serial communications based on a 4-pin interface. Can be used either in master or slave mode*
I2C (x2)	**I²C bus** *Compliant with the I2C bus specification, version 2. Supports master and slave modes*
CAN (x2)	**Controller Area Network** *Bus interface controller compliant with ISO 118980-1, CAN 2.0A and CAN 2.0B standards*
UART (x2)	**Universal Asynchronous Receiver Transmitter** *Low rate data modem interface for serial communication. Often used for Terminal connections to a host PC*
GPIO	**General Purpose Input/Output** *There are four banks GPIO, each of 32 bits*
SD (x2)	For interfacing with **SD card** memory
USB (x2)	**Universal Serial Bus** Compliant with USB 2.0, and can be used as a host, device, or flexibly (*on-the-go*, or OTG mode, i.e. can switch between host and device modes)
GigE (x2)	**Ethernet** Ethernet MAC peripheral, supporting 10, 100, and 1Gbps modes

PS-PL interfaces

- Based on either:
  - **Advanced eXtensible Interface** (current version is AXI4, part of the ARM AMBA® 3.0 open standard)
  - or **Extended Multiplexed Input/Output** (EMIO)
- AXI: three different protocols:
  - **AXI4:** memory-mapped, data burst transfer of up to 256 data words (or "data beats")
  - **AXI4-Lite**: memory-mapped, only one data transfer per connection (no bursts)
  - **AXI4-Stream**: no address, unrestricted-size burst transfers (streams)
- Interfaces
  - point-to-point connections
  - master/slave ports

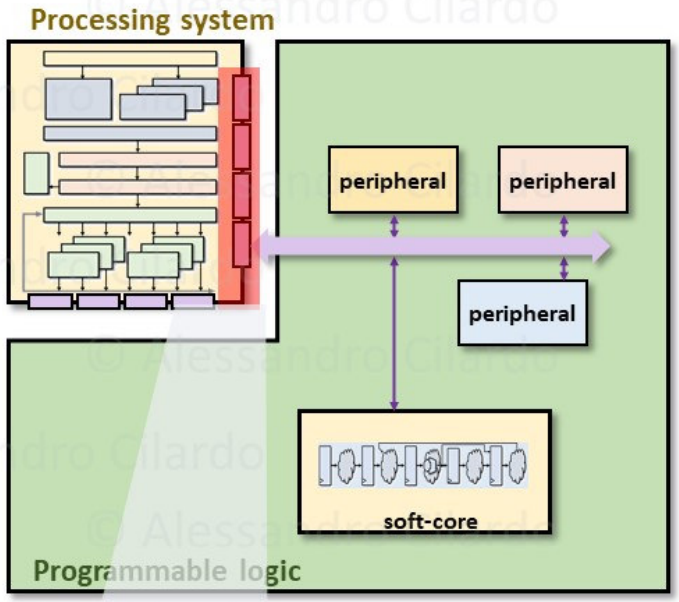


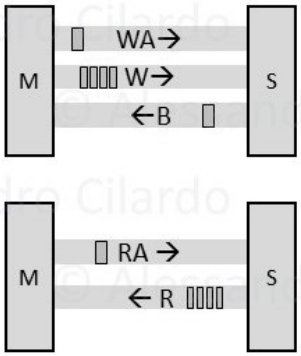




PS-PL interfaces

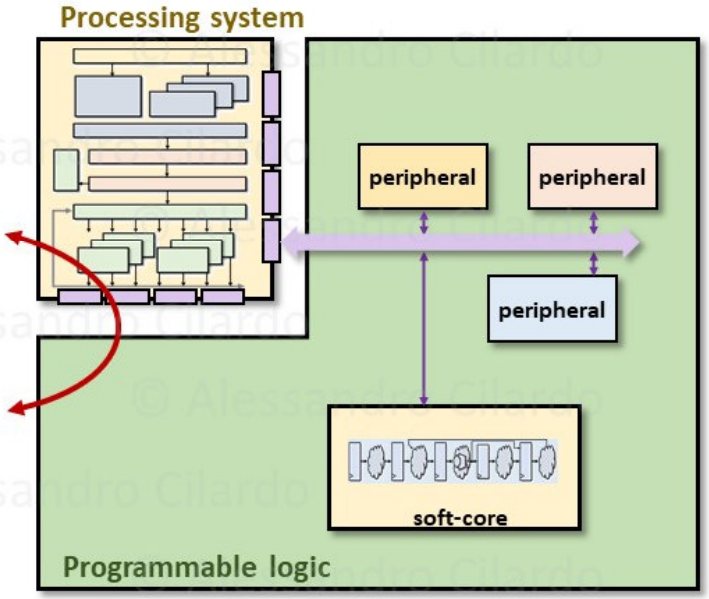
- General Purpose AXI
  - 32-bit data bus, suitable for low and medium rate PL-PS communications
  - PS is master of two interfaces, PL of the other two
  - doesn't include buffering
- Accelerator Coherence Port
  - PL-SCU within the APU, 64-bit bus width
  - achieves coherence between the APU caches and PL
- High Performance Ports
  - FIFO buffers for "bursty" read and write operations
  - high-speed communication, 32- or 64-bit data width
  - PL is master of all four interfaces
  - Relies on the AMBA4 AXI full standard + 1kB FIFO for each port





Extended Multiplexed Input/Output (EMIO) Interfaces

- Involve signal transfer between PS and PL
- EMIO can provide additional 64 inputs / 64 outputs
- Another option is to use EMIO to interface the PS with a peripheral block in the PL
- Other signals crossing the PS-PL boundary include
  - watchdog timers
  - reset signals
  - interrupts
    - 16 peripheral interrupts from PL to PS
      - used for accelerators and peripherals in PL
    - 4 processor-specific interrupts from PL to PS
    - 28 interrupts from PS peripherals to PL
      - PS peripherals can be serviced from Microblaze in fabric
  - and DMA interfacing signals

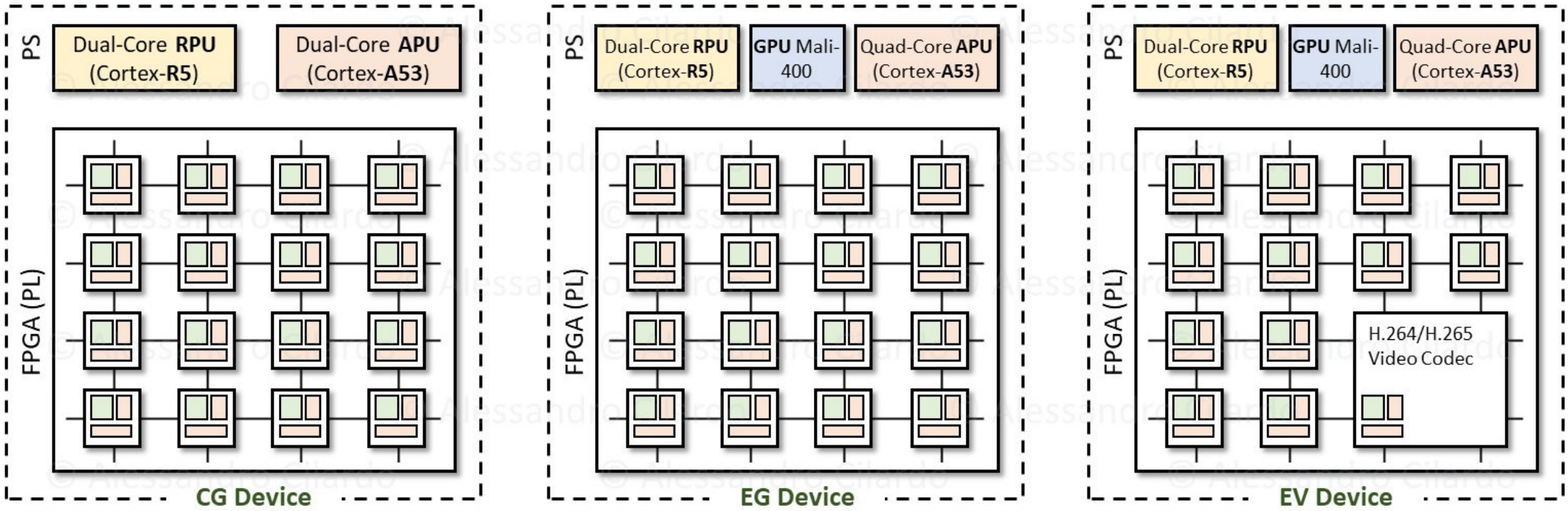


Cortex-A53 based Xilinx Zynq Ultrascale+ MPSoC

- Zynq MPSoC (2015)
  - Dual or Quad APU, Dual RPU (opt. an Arm Mali GPU), General Purpose and Video domains
  - PS: Arm Cortex-A53 for APU, Cortex-R5 for RPU, VCU IP supporting H.265 and H.264
  - PL: Xilinx Ultrascale+, up to 1M+ Flip-Flops and 500k+ LUTs
  - PCIe Gen2, USB3.0, SATA 3.1, DisplayPort, Gigabit Ethernet, ..
  - Configuration and Security Unit, Platform Management Unit, ..
  - PS Frequency: up to 1.5GHz; PL Frequency up to 891GHz
- Application domains:
  - mobile base-station signal processing, video compression/decompression, broadcast camera equipment, navigation and radar systems, high speed switching, routing infrastructure for data centres, Advanced Driver Assistance Systems (ADAS), and even big data analytics, ..

ARM Cortex subsystem within the Zynq MPSoC

- Arm Intellectual Property (IP) licensed to Original Equipment Manufacturers (OEMs) such as Xilinx:
  - Cortex-A53 and Cortex-R5 processor + additional Arm IPs in the MPSoC
  - can be partly customized by the OEM



Processing System (PS) in Zynq MPSoC: bird's eye view

- Three distinct subsystems
  - Cortex-A53 MPCore, Governor, Level 2 Memory System
  - Governors: provides clock, debug, and tracing resources to respective cores
- System interconnect:
  - Arm Cache Coherent Interconnect (CCI)
- Dual-core Cortex-R5 based Real-time Processing Unit (RPU)
- Arm Mali-400 MP2 GPU (EG and EV)
  - support for OpenGL and Open VG
- General-purpose connectivity
  - Multiplexed Input/Output (MIO)
- High-speed connectivity
  - multi-gigabit transmit and receive channel pair (PS-GTR transceiver)
- Platform Management Unit (PMU), Configuration Security Unit (CSU), and Battery Power Unit (BPU), ..

Application Processing Unit and memory subsystem

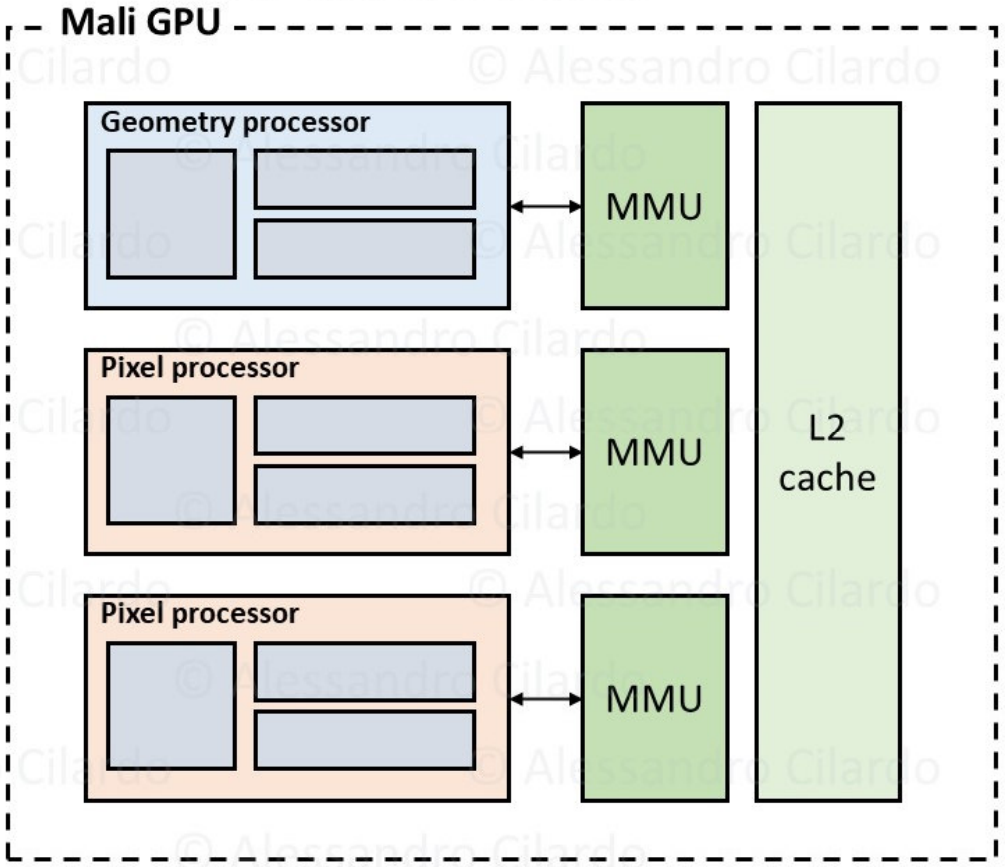
- Arm Cortex-A53 Multi-Processor Core
  - up to 1.5 GHz, supports both 32- and 64-bit ISA
- Floating Point Unit (FPU)
- NEON Media Processing Engine (MPE)
- Cryptography Extension (Crypto)
- Memory Management Unit (MMU)
- Private 32 KB Level 1 cache per core (instructions/data)
- 1MB Level 2 shared cache memory
- Snoop Control Unit (SCU)
  - allows internal cache-to-cache transfers and transactions between the APU and PL via ACP
  - cache coherence relies on a MOESI protocol
- L3 memory (off-chip DDR, on-chip OCM, PL memory blocks)
- 128-bit AXI Coherency Extension (ACE)
  - hardware-maintained coherence
- Generic Interrupt Controller (GIC)

Interrupts

- Many different interrupt sources
  - processor cores and other IPs in the PS, custom cores in the PL, ..
- Interrupts managed by the Generic Interrupt Controller (GIC)
  - handles interrupt sources as well as advanced aspects like security and virtualization
  - the Zynq MPSoC uses the Arm *GICv2* architecture.
- GIC supports four types of interrupts:
  - peripheral interrupts (either private to a processor or shared across the MPSoC)
  - Software-Generated Interrupts (SGIs)
    - rely on a particular GIC register, are used for inter-processor communication
  - virtual interrupts
    - generated by a hypervisor collecting and distributing physical interrupts, including software ones
  - maintenance interrupts for virtual machines, used for managing interrupt behaviors
- A "distributor" (part of the GIC) applies interrupt priorities and states:
  - lower indices→higher priority. States: inactive, pending, active, active/pending

Graphics Processing Unit

- Arm Mali-400 MP2 GPU
- Two-dimensional (2D) and three-dimensional (3D) graphic acceleration
- One geometry processor (GP) and two pixel processors (PP)
- up to 667 MHz
- OpenGL ES 1.1/2.0 and OpenVG 1.0/1.1 API support
- 64KB Level 2 cache with an Advanced Peripheral Bus (APB) slave



Communication across PS and PL

- Main components within the PS:
  - APU and Snoop Control Unit
  - Cache Coherent Interconnect (CCI)
  - system memory management unit (SMMU)
  - central switch and low-power switch
- PS is physically connected to the PL
  - PS-PL interfaces can either be Masters or Slaves (as seen from PS), and are part of Low-Power Domain or Full-Power Domain
- Three types of coherent ports
  - ACP (ACE-Lite for L1 and L2 APU caches)
  - AXI ACE (connected to CCI)
  - and HP ports going through SMMU
- PL cores can access the PS memory space
  - the system MMU (SMMU) provides address translation for access to virtual addresses from the PL

External interfaces and peripherals

- Multiplexed Input/Output (MIO)
- 78 of the PS I/Os are mapped through MIO
- High-speed (multi-Gb/s) serial connectivity
  - Serial Input Output Unit (SIOU)
  - relies on the PS-GTR transceiver (up to 6Gb/s)
  - supports PCIe, USB 3.0, DisplayPort, SATA and Ethernet interface protocols
  - USB3.0 and Ethernet peripheral interface blocks shared with MIO
- Extended MIO (EMIO)
  - accessed by the PL

UART (x2)
SPI (x2)
CAN (x2)
I2C (x2)
GPIO
GigE (x4)
NAND x8 and ONFI 3.1
USB 3.0 (x2)
SD/eMMC (x2)
Quad-SPI x8

Multiplexed Input/Output (MIO) Peripheral Interfaces

DisplayPort
USB 3.0 (x2)
SATA (x2)
PCIe
GigE (x4)

High-Speed Connectivity I/O Peripheral Interfaces