A SoC Case Study

Module 11

Module Syllabus

- System-on-Chip (SoC) concepts
- Overview of typical IP blocks
- Example SoC designs

A System-on-chip (SoC) Case Study

- In this final module, we are going to explore a modern System-on-Chip or “SoC.”
  - SoCs are designed for particular markets and meet target design goals by combining many different IP blocks.
  - They contain both programmable processors and application-specific IP.
- We’ll examine each of its building blocks and main features.
- And look at how each of these individual “IP” blocks are brought together

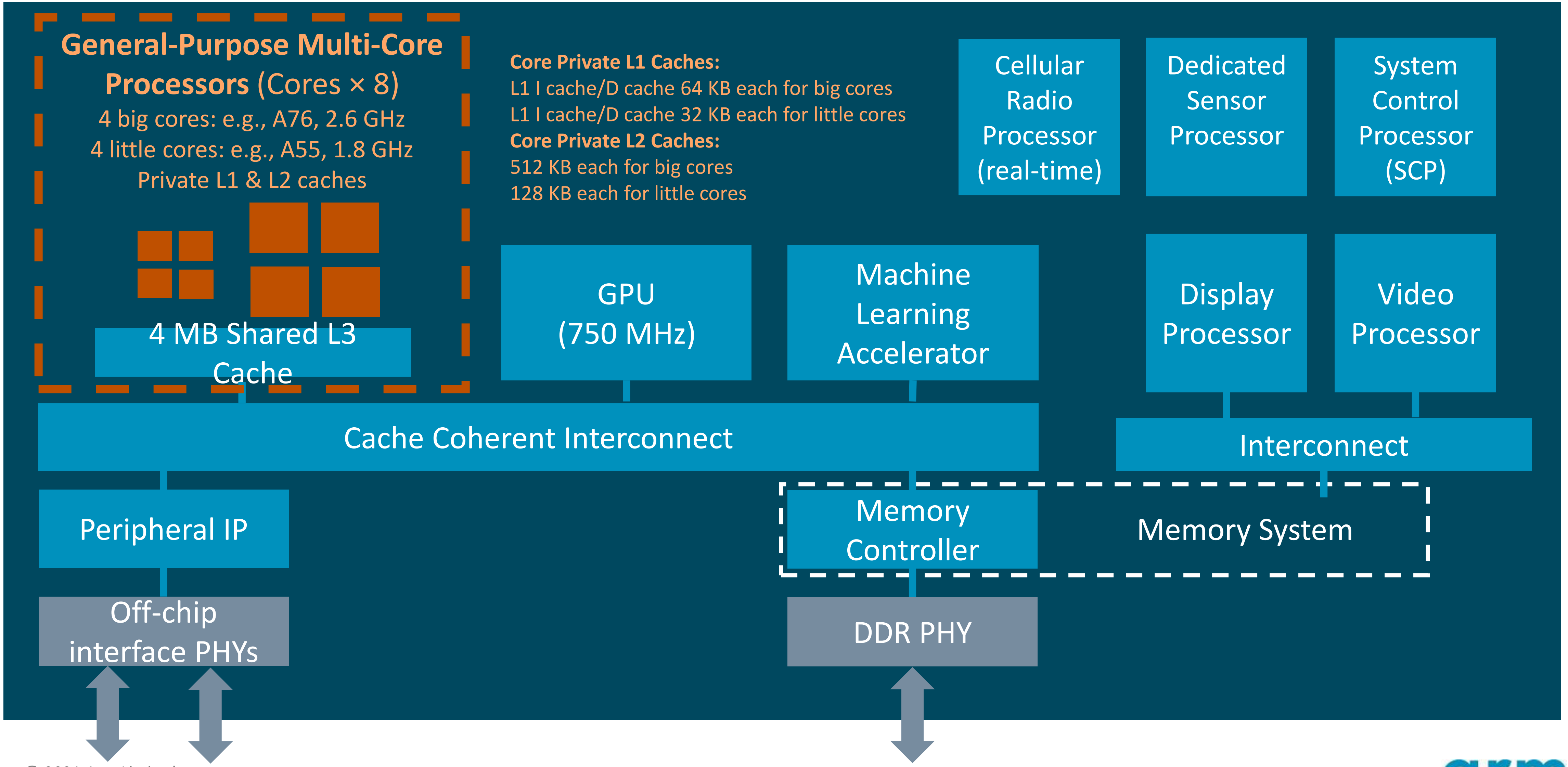
Why Create a Custom System-on-chip (SoC)?

- Reduced cost
- Reduced PCB area and volume
- Increased performance and reduced power consumption
- Product differentiation
- SoCs integrate a range of IP types: processors, custom processors, accelerators, on-chip memories, peripherals and interfaces, etc.

System-on-chip (SoC) Design

- Let’s define some high-level design goals and constraints for a SoC example:
- **Target market**: client device (e.g., mobile phone, tablet, etc.)
- **Area** = ~70-80 mm^2 in 7 nm fabrication technology
- **Transistor budget**: ~7-8 billion transistors
- **Performance:** Excellent general-purpose performance over a wide range of workloads, e.g., image processing, 2D/3D graphics, machine learning applications. Some requirements for real-time processing capabilities
- **Off-chip memory bandwidth:** 32 GB/s
- **Power:** 2-3 W peak (only in short bursts on smartphones)

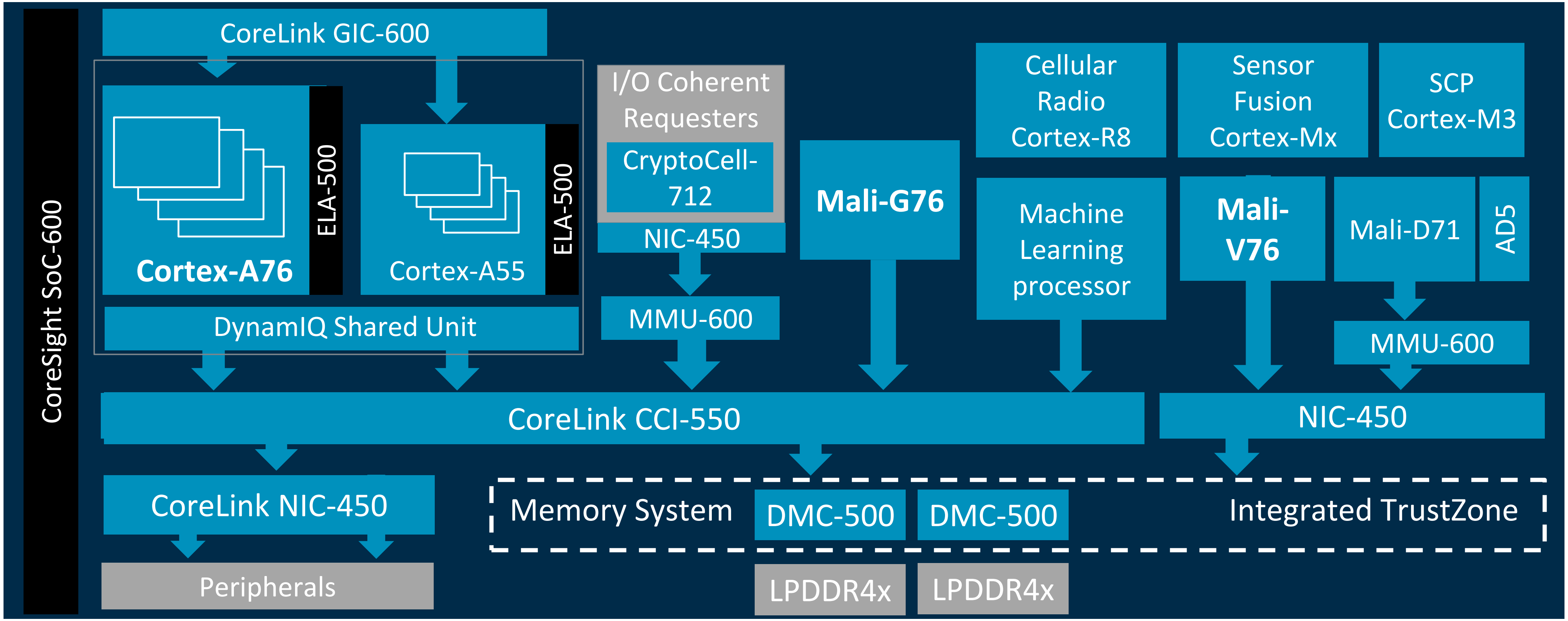
High-level View of Our System-on-chip (SoC) Example



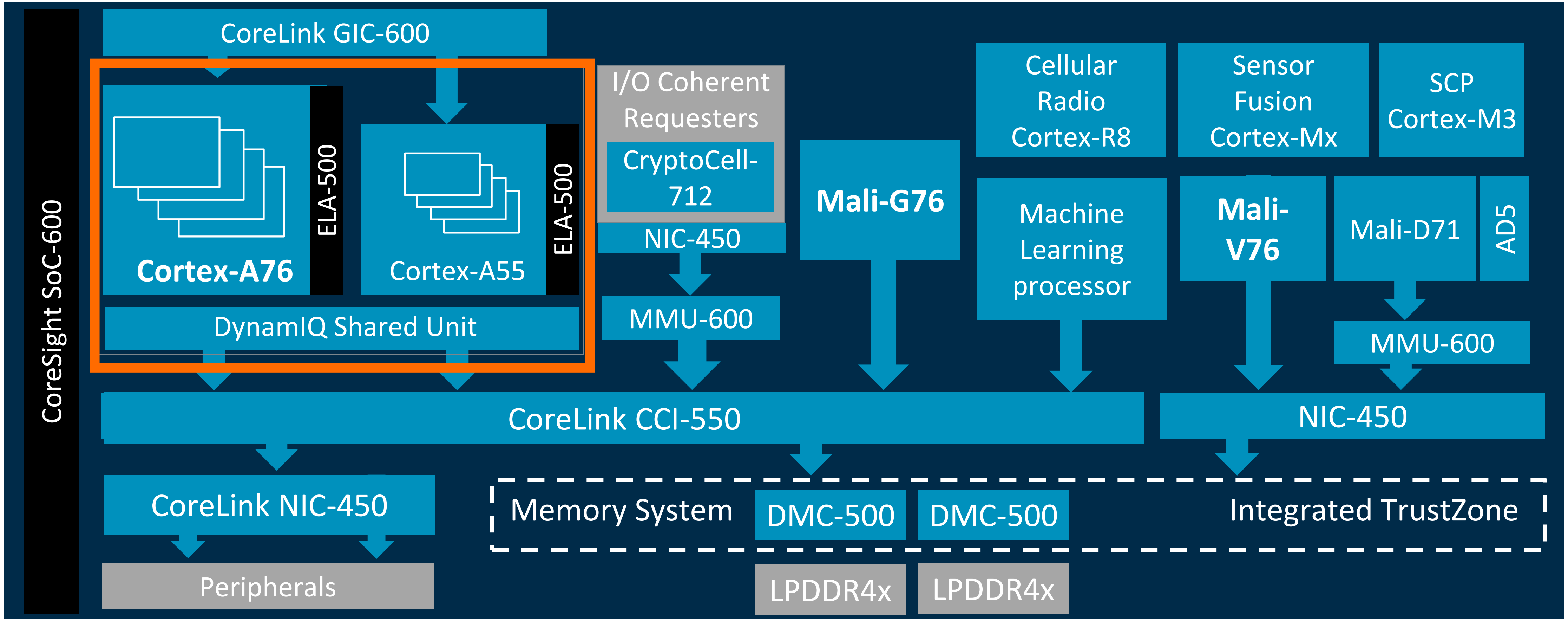
General-Purpose Multi-Core Processors (Cores × 8)
4 big cores: e.g., A76, 2.6 GHz
4 little cores: e.g., A55, 1.8 GHz
Private L1 & L2 caches
4 MB Shared L3 Cache
Core Private L1 Caches:
L1 I cache/D cache 64 KB each for big cores
L1 I cache/D cache 32 KB each for little cores
Core Private L2 Caches:
512 KB each for big cores
128 KB each for little cores
Cellular Radio Processor (real-time)
Dedicated Sensor Processor
System Control Processor (SCP)
GPU (750 MHz)
Machine Learning Accelerator
Display Processor
Video Processor
Cache Coherent Interconnect
Interconnect
Peripheral IP
Memory Controller
Memory System
Off-chip interface PHYs
DDR PHY

32 GiB/s (LPDDR4X-4266, 4 × 16-bit channels)

A Modern Arm System-on-chip (SoC)



Heterogeneous Multicore Cluster



Heterogeneity

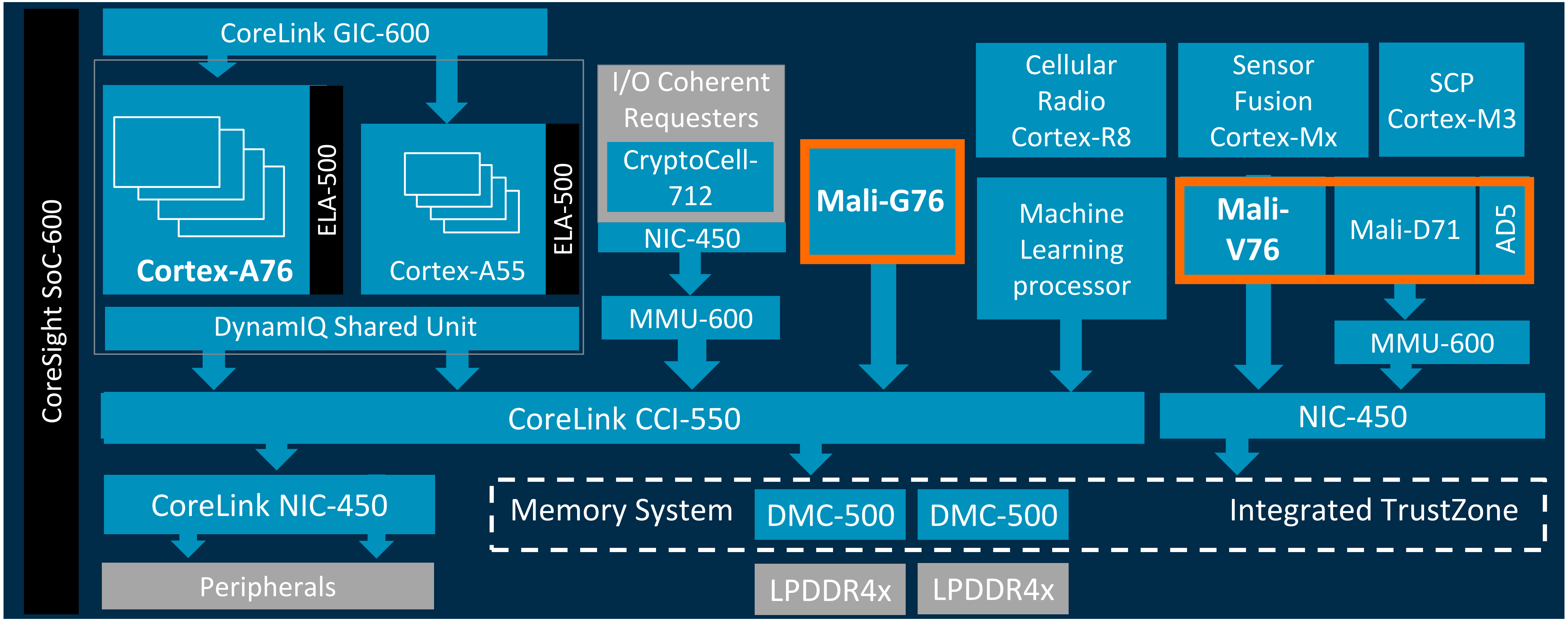
- There are different levels of heterogeneity within the SoC.
- Cores running the same ISA but with different microarchitectures
  - E.g., Cortex-A55 and Cortex-A76 in DynamIQ Shared Unit
  - Allows general-purpose tasks to migrate to save power or increase performance
- Cores extracting different types of parallelism
  - E.g., Cortex-A cores vs Mali GPU
  - The GPU is specialized to efficiently exploit data-parallel parallelism.
- Cores specialized to specific tasks
  - E.g., machine-learning processor
  - These are highly specialized hardware accelerators designed for a narrow range of workloads.

Key aim: Reduce power consumption but increase performance

Heterogeneity in Microarchitectures

- An example is DynamIQ big.LITTLE.
  - Next generation big.LITTLE Cortex-A CPUs in one cluster with a shared coherent memory
  - Tasks that do not require high performance can migrate to the smaller cores (Cortex-A55).
  - Tasks that need computing power run on the larger cores (Cortex-A76).
- The key to this form of heterogeneity is ISA-compatibility.
  - Both core types must run exactly the same ISA so as to enable migration at any point.
- Shared L3 caches and separate voltage/frequency domains within each cluster.
  - These improve performance and provide more opportunities for power saving.

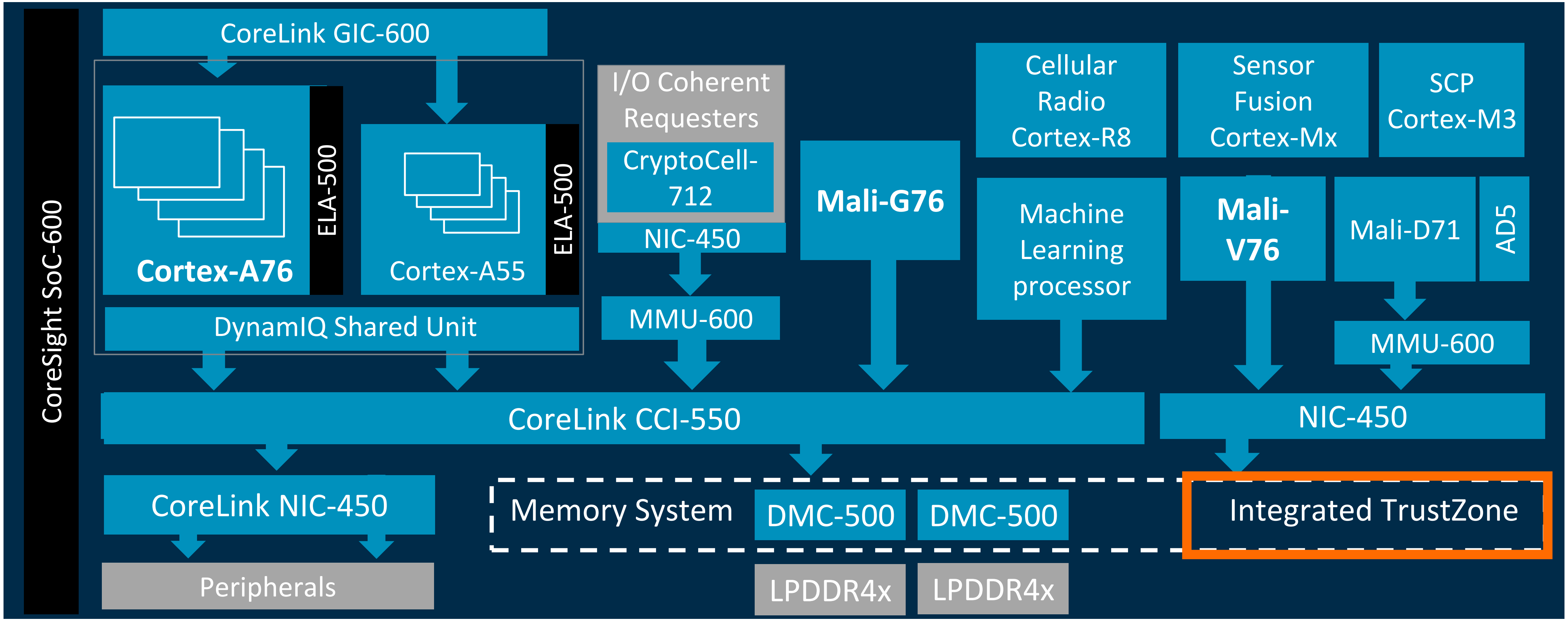
Graphics and Display IP



Graphics and Display IP

- In the example shown in the previous slide:
- **Mali-G76 (see module 10)**
  - This is the SoC’s main GPU. Our SoC has 10 GPU cores providing 240 32-bit execution lanes (with INT8 support).
- **Mali-V76**
  - Video processor (video encode and decode)
- **Mali-D71**
  - Display processor (scaling, rotation, composing layers, picture quality enhancements)
- **Assertive Display 5 (AD5)**
  - HDR management features
  - Ambient light adaptivity and advanced power-saving features
  - Gamut management

TrustZone


CoreSight SoC-600
CoreLink GIC-600
Cortex-A76
ELA-500
Cortex-A55
ELA-500
DynamIQ Shared Unit
I/O Coherent Requesters
CryptoCell-712
NIC-450
MMU-600
Mali-G76
Cellular Radio Cortex-R8
Sensor Fusion Cortex-Mx
SCP Cortex-M3
Machine Learning processor
Mali-V76
Mali-D71
AD5
MMU-600
CoreLink CCI-550
NIC-450
CoreLink NIC-450
Peripherals
Memory System
DMC-500
DMC-500
LPDDR4x
LPDDR4x
Integrated TrustZone

What Is Security?

- Security is a property of the system, which ensures that resources of value cannot be copied, damaged, or made unavailable to genuine users.
- There are several fundamental security properties:
- Confidentiality
  - If an asset is confidential, it cannot be copied or stolen by a defined set of attacks.
- Integrity
  - If an asset has its integrity assured, it is defended against modification by a defined set of attacks.
- Authenticity – may be provided if integrity cannot be
  - If an asset is authentic, it is known to have not been modified by an attacker.
  - In other words, the defender can detect any modifications made before the asset is used.

TrustZone

- TrustZone's primary security objective:

To enable the construction of a programmable environment that allows the confidentiality and integrity of almost any asset to be protected from specific attacks

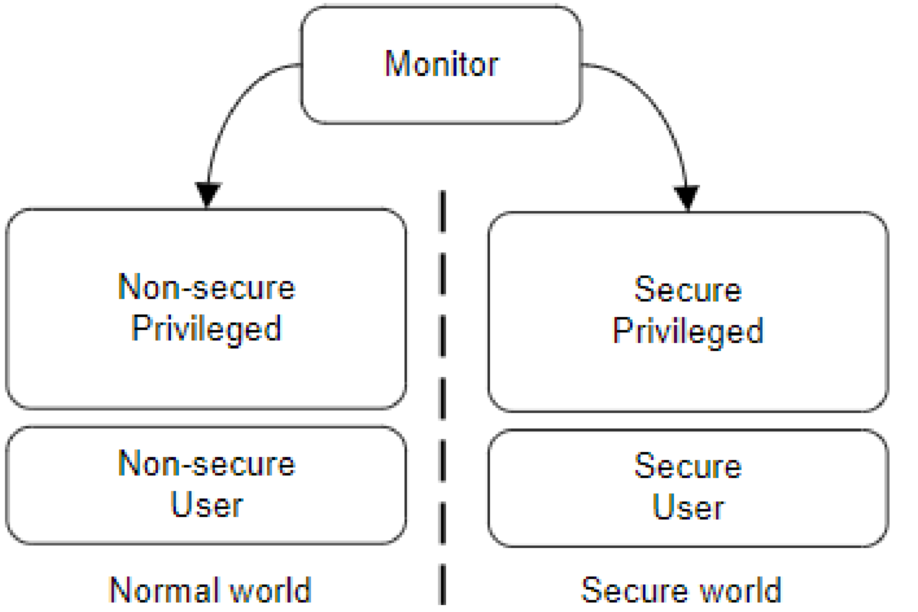
- This is achieved by partitioning all resources into two worlds:
  - The secure world for the security subsystem
  - The normal world for everything else
- No normal world components can access secure world resources.
  - This is enforced through hardware logic.

Hardware Requirements

Reducing hardware overheads

- Use two cores to implement two worlds.
  - I.e., a dedicated core for the Secure world
  - Costly in silicon area and power
- TrustZone provides architectural extensions to allow one core to execute code from the normal and secure worlds.
- "Secure Monitor Mode" acts as a gatekeeper for moving between worlds.

Secure Monitor

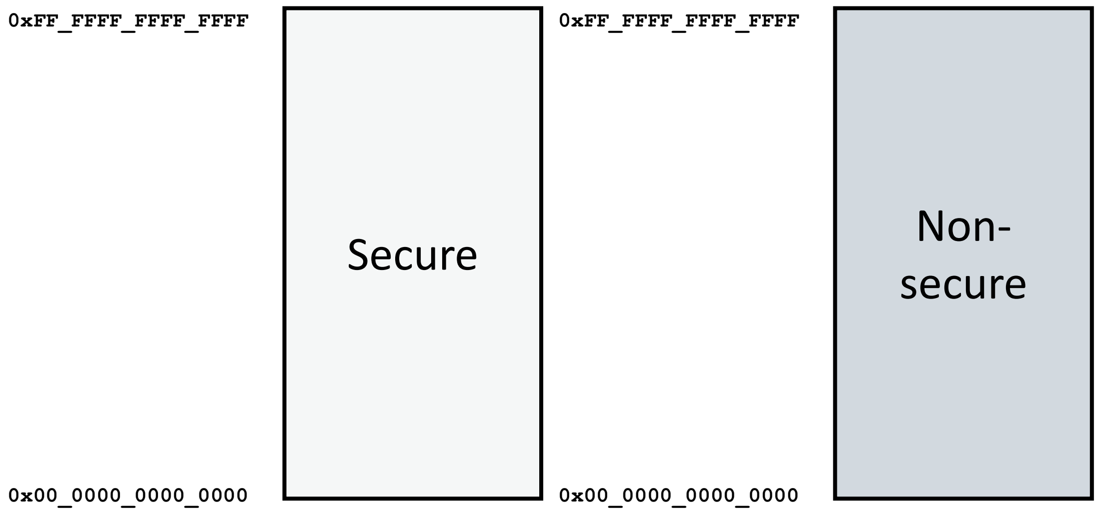


TrustZone Processor Architecture

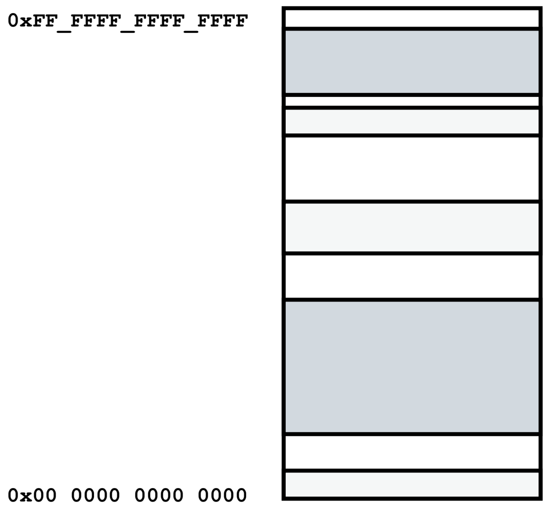
- A core executes code from both Secure and Normal (Non-Secure) worlds by presenting two virtual processors to the outside world:
  - The non-secure virtual processor can only access Non-Secure resources.
  - The secure virtual processor can see all resources.
- Virtual processors share the core through time slicing (see Module 9).
  - They context switch through a dedicated core mode that performs the switch.
  - One example is the Secure Monitor Call (SMC) instruction.
- The secure monitor ensures all secure world state is inaccessible when switching to normal mode.

Memory Partitioning

- Physical memory map is also partitioned into two worlds, i.e., Secure and Non-Secure regions.
  - Virtual addresses in Non-Secure state can only map to Non-Secure physical addresses.
  - Virtual addresses in Secure state can map to either Secure or Non-Secure physical addresses.
- Memory transactions have a security attribute to indicate Secure or Non-Secure access.

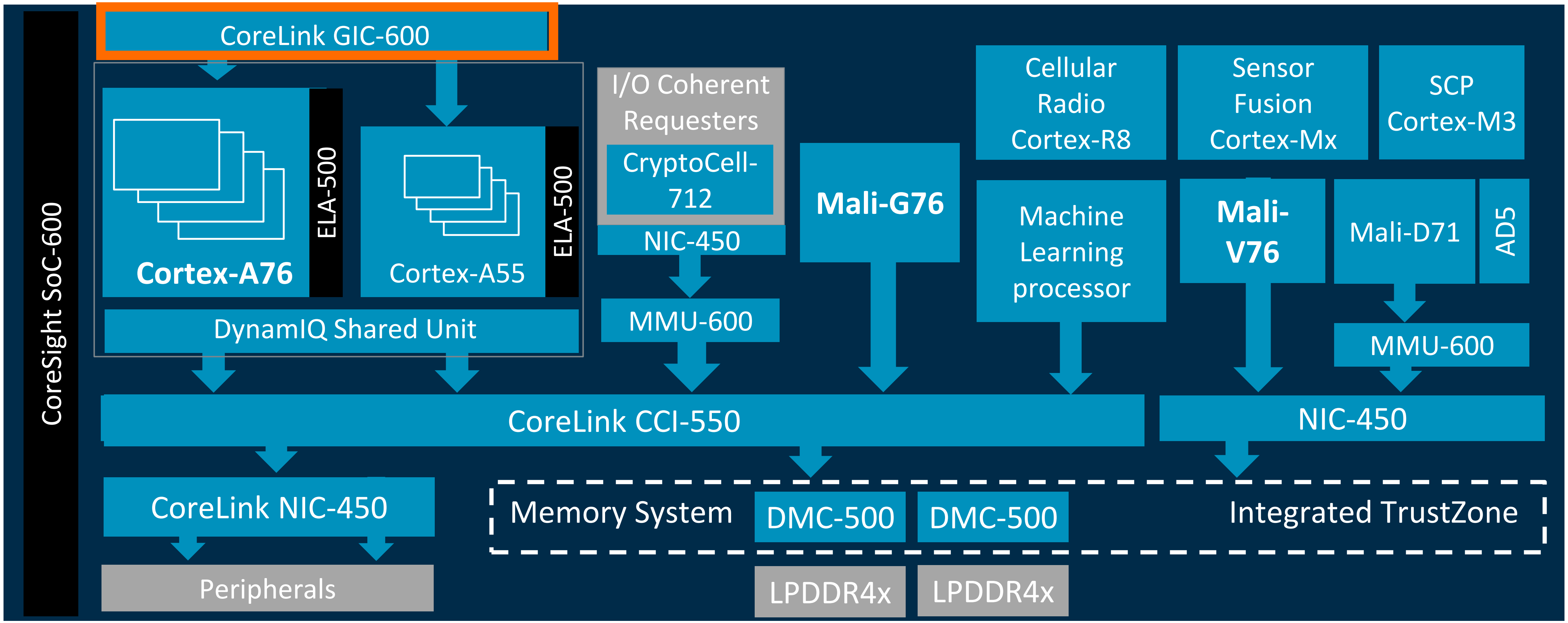


Architectural view on the physical address space
One physical address space for each world



Common implementation
Data are interleaved throughout memory

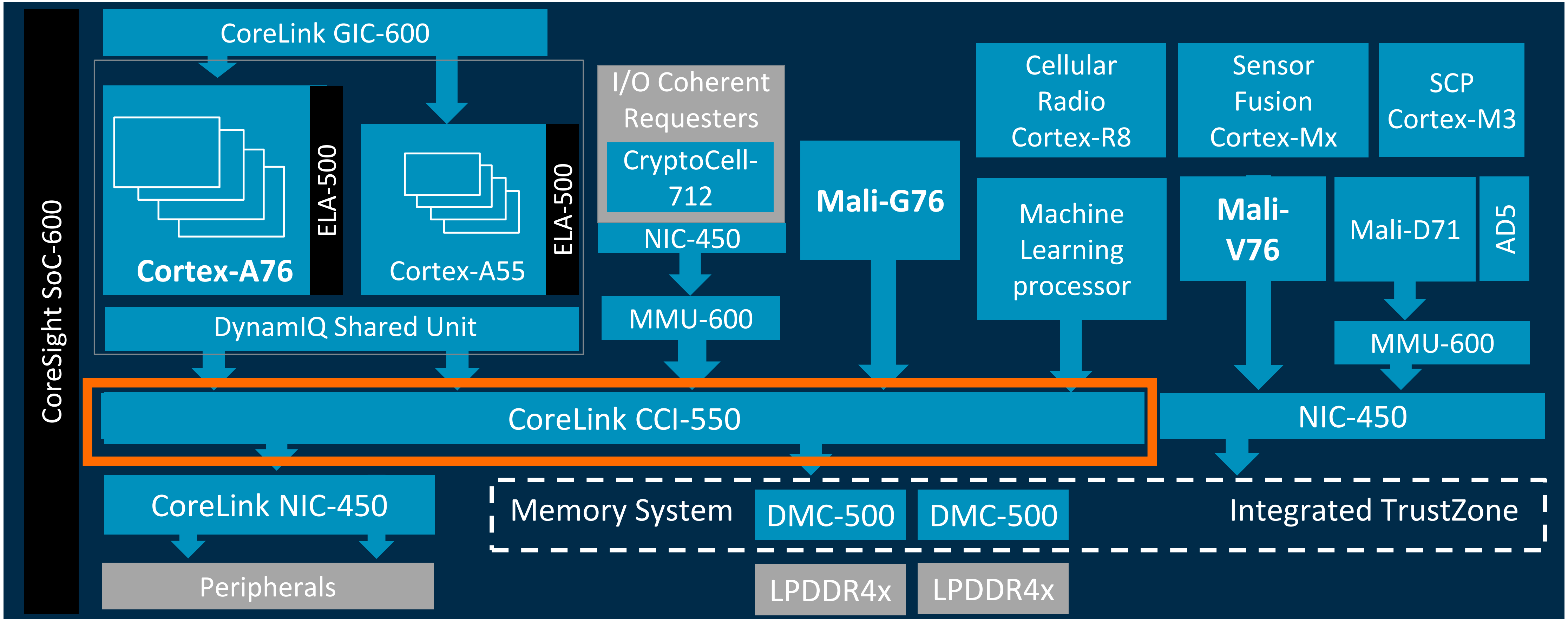
Generic Interrupt Controller



Generic Interrupt Controller

- Performs interrupt management, prioritization, and routing
- Boosts processor efficiency and interrupt virtualization
- Arm GIC architecture
- A fully coherent GPU simplifies software development and improves performance.
  - Removing the need for software-managed cache maintenance
- CoreLink CCI-600 Generic Interrupt Controller
  - Supports DynamIQ cores
  - Fully backward compatible with Arm v8.0 cores

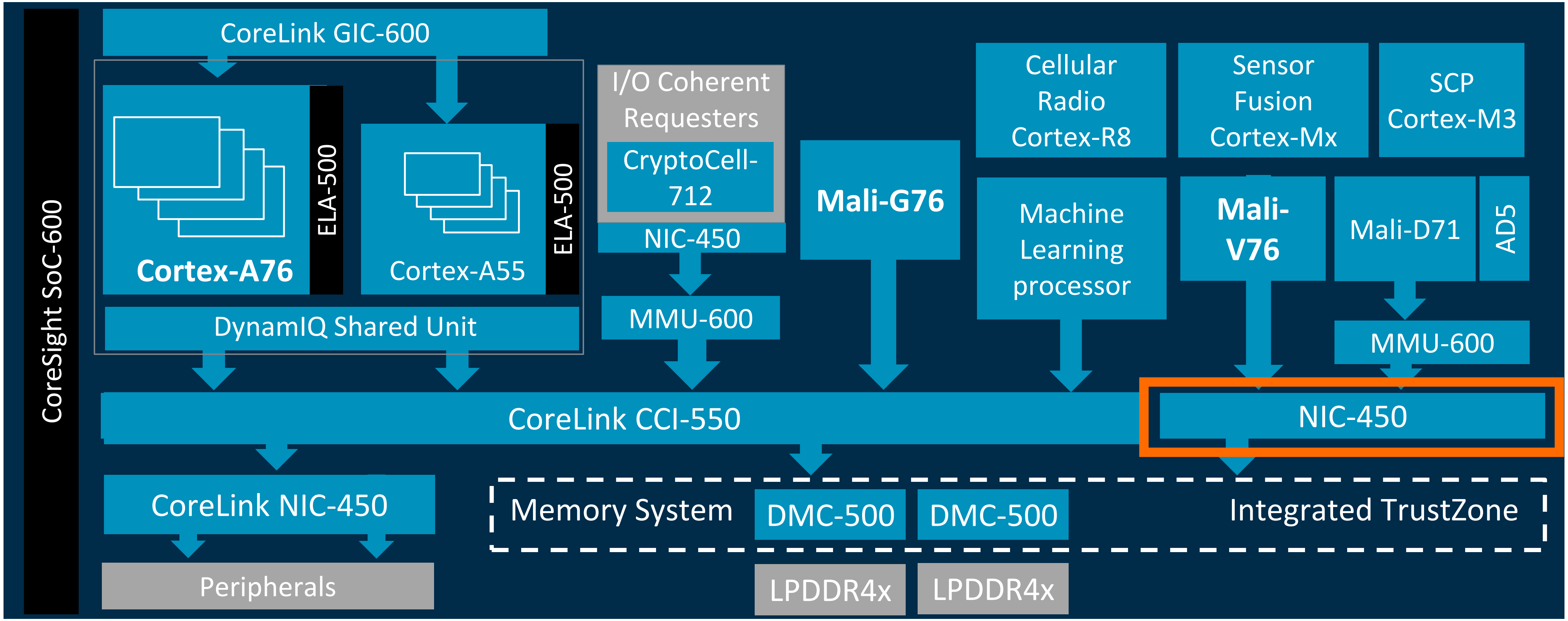
Interconnect



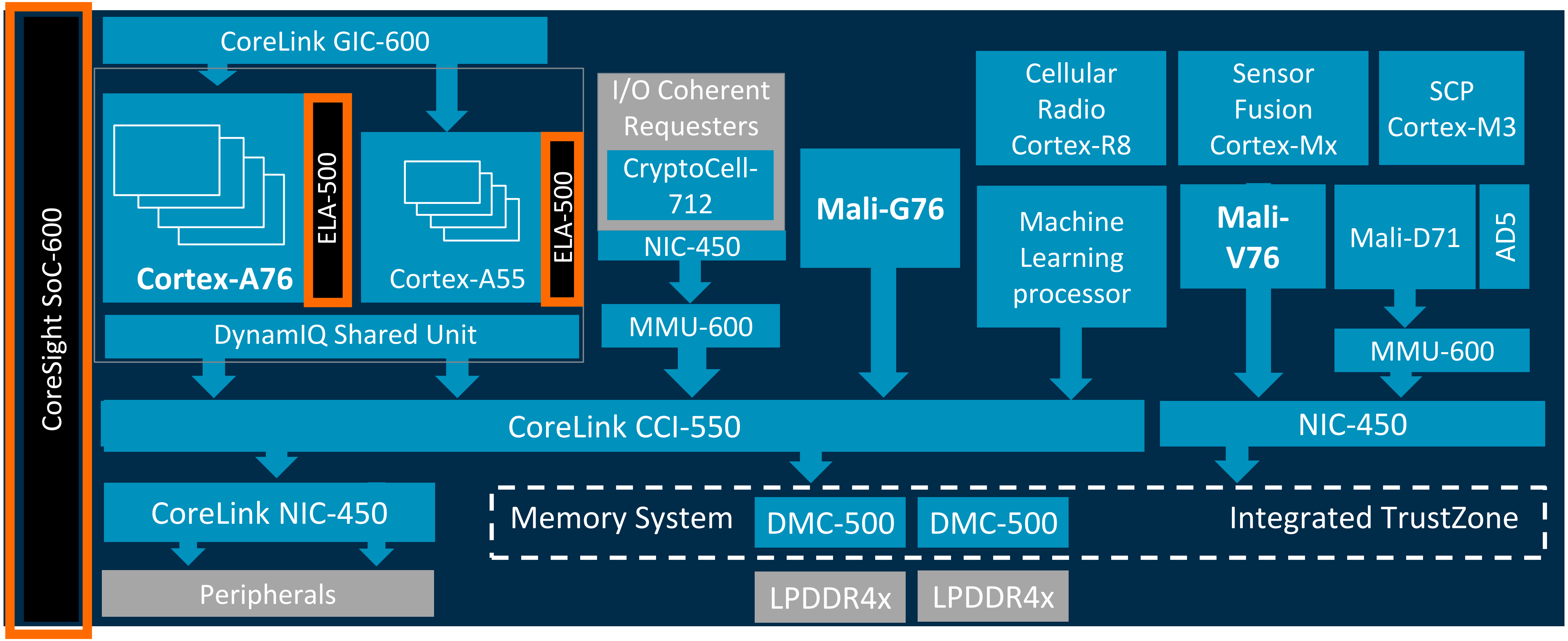
Cache Coherent Interconnect

- CoreLink CCI-550 Cache Coherent Interconnect
- Provides cache coherence between CPU clusters and the GPU, network interfaces, and the machine learning accelerator
- A fully coherent GPU simplifies software development and improves performance.
  - Removing the need for software-managed cache maintenance

Interconnect



Debug and Trace

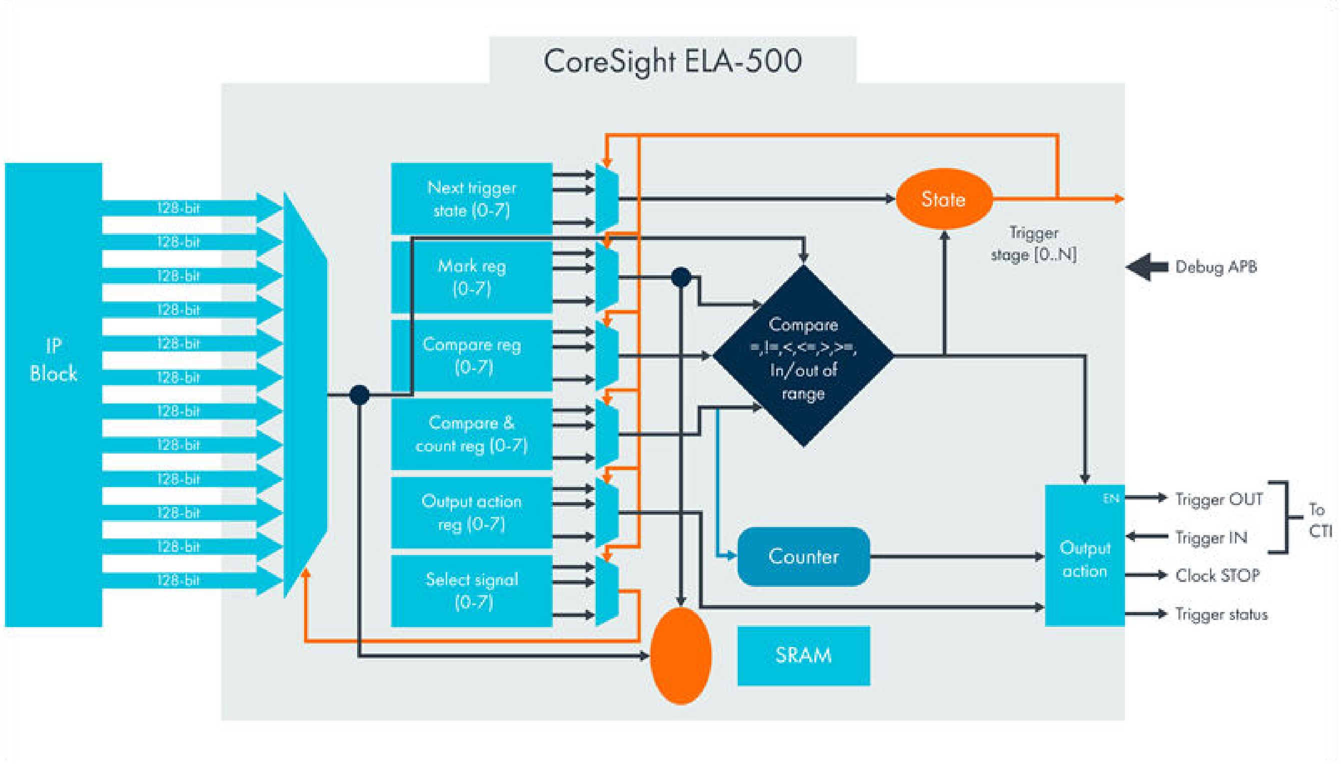


Debug and Trace

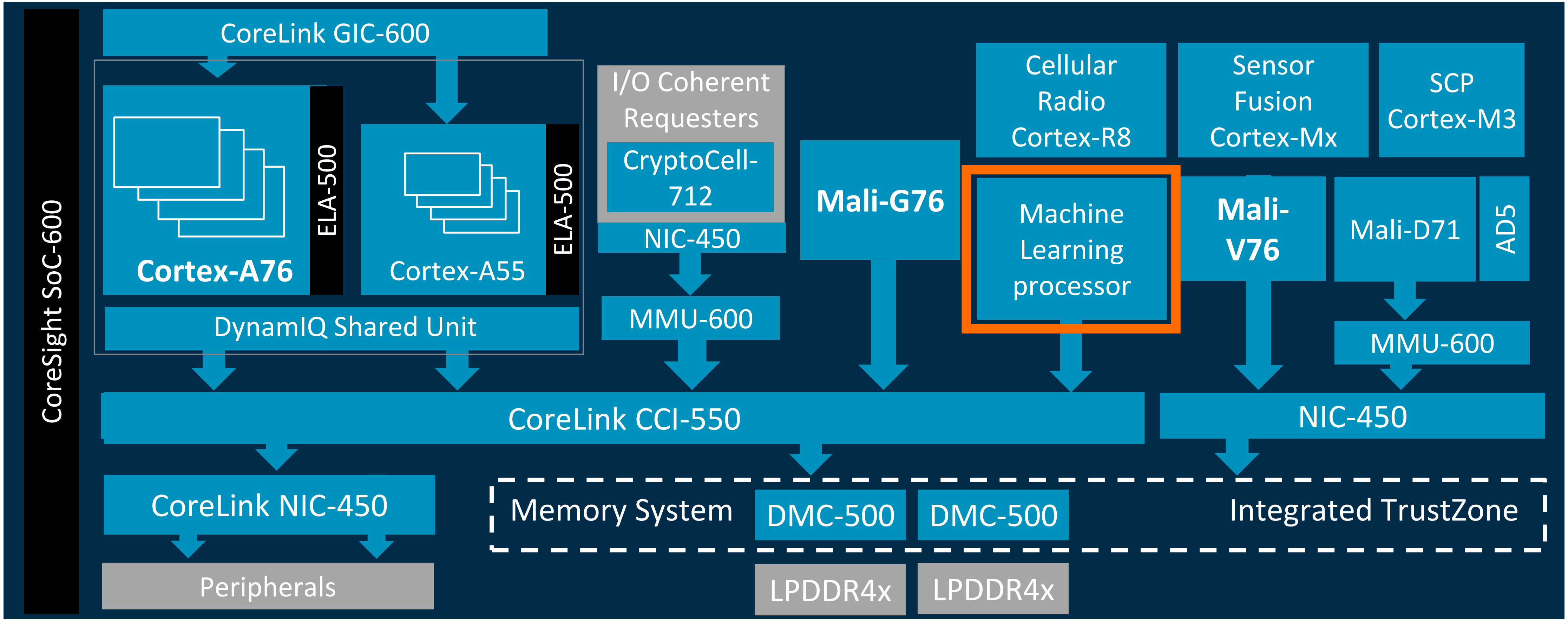
- **Traditional debug** (CoreSight SoC-600)
  - Invasive debug where the processor is halted using breakpoints or watchpoints
  - A debug connection is used to examine and modify registers and memory.
  - It is possible to “single step” execution.
- **Trace** (CoreSight SoC-600)
  - Non-invasive debug with the processor running at full speed
  - Delivers the trace off-chip in real-time or stores it in on-chip memory
  - Traces are usually carefully compressed to make the best use of limited off-chip bandwidth (or on-chip memory).
  - Various components can create traces.

Debug and Trace

- **Embedded Logic Analyzer** (CoreSight ELA-500)
  - These IP blocks allow many (over 1500) internal logic signals to be monitored.
  - Complex “triggers” can be constructed to start capturing signals.
  - Signals are stored in on-chip SRAM for later analysis.



Specialization

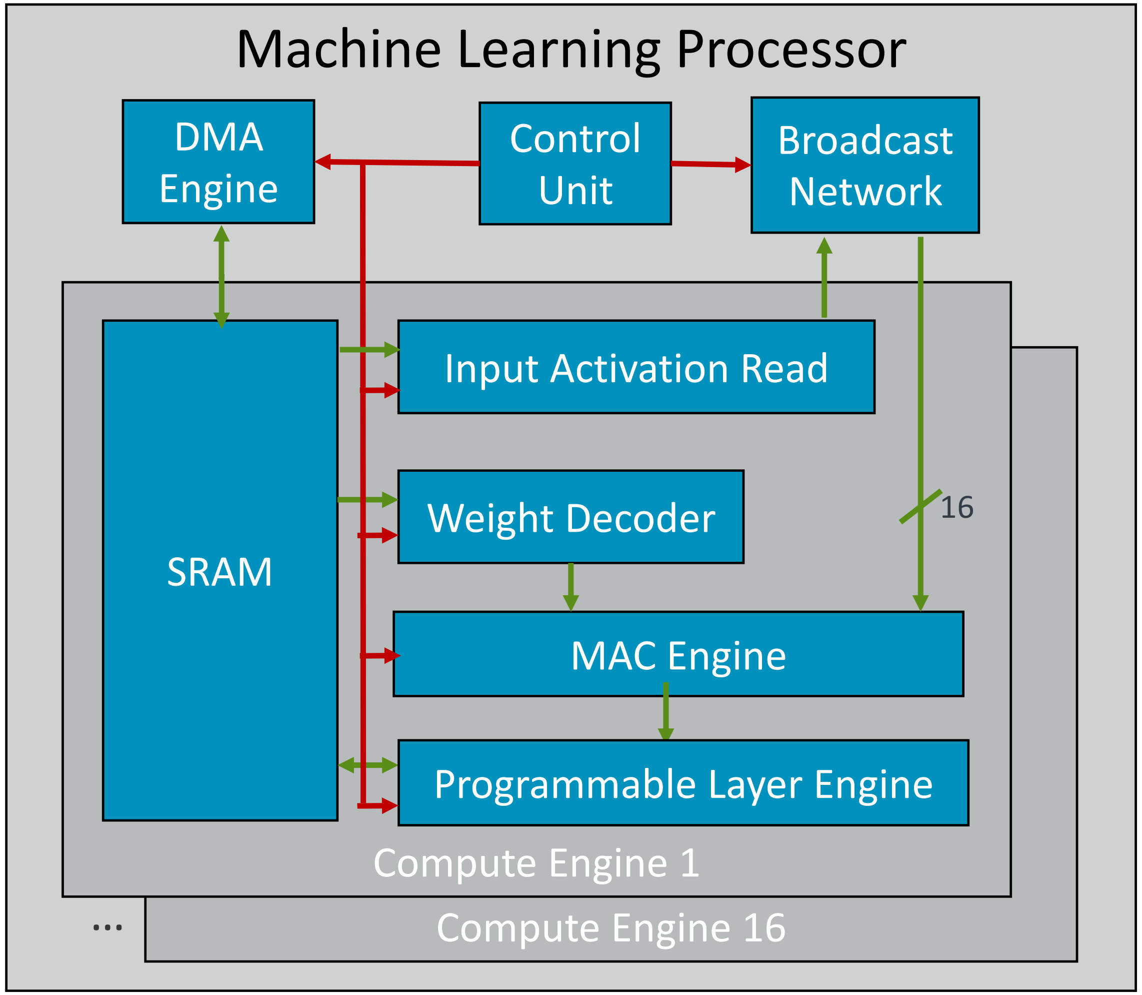


Specialization – A Machine Learning Processor

- A Machine Learning (ML) processor’s architecture is different to general-purpose cores.
  - Different instructions
  - Different programming model
  - Different pipeline structure
  - Different methods of communication
- The ML processor executes convolutional neural-network inference efficiently.
  - Design and execution tailored to the specifics of the algorithm

Specialization – A Machine Learning Processor

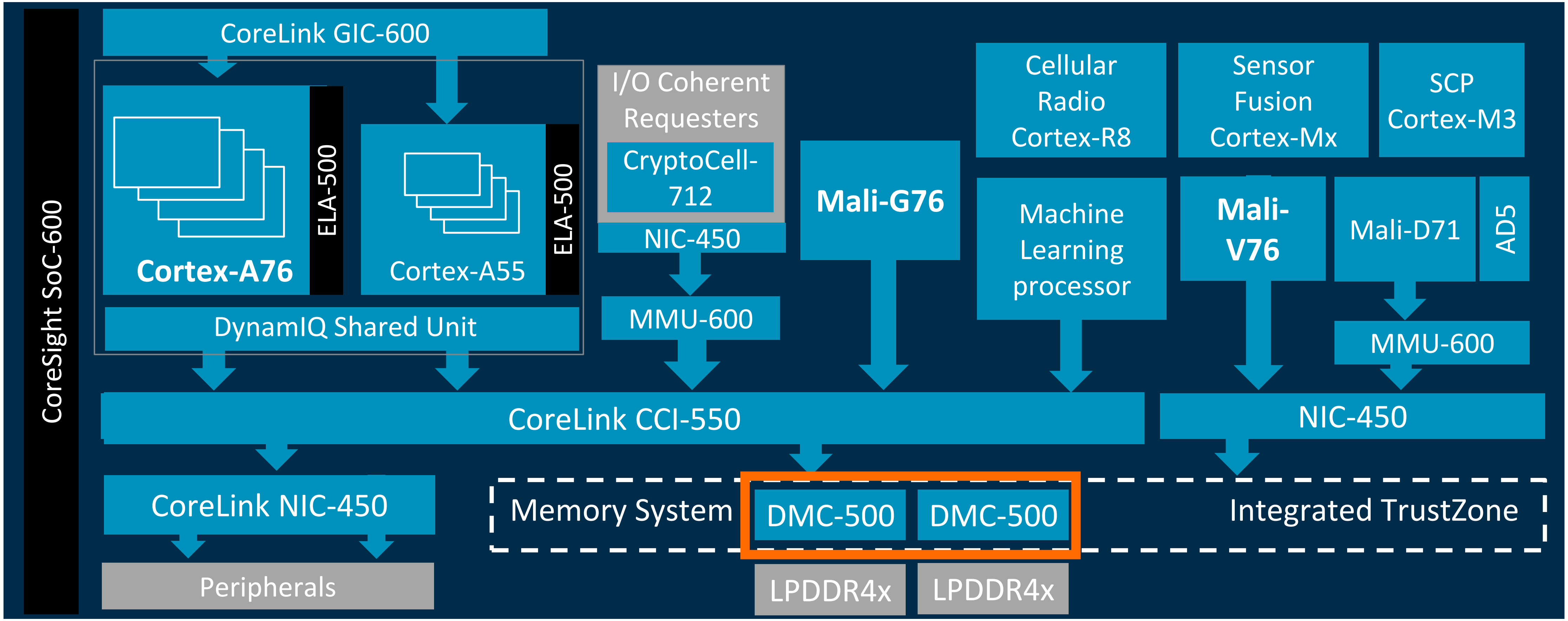
- Specialized for neural-network inference
- Example of Arm ML processor with 16 compute engines
  - 8-bit quantized integer support
  - No caches
- Convolutional neural networks statically mapped onto the compute engines
  - Output feature maps interleaved across engines
  - Weights held in that engine's SRAM
  - Input feature maps interleaved across all SRAMs



Specialization

- Specialization and heterogeneity are interlinked.
  - We can think of the Cortex-A55s as being “specialized” for low-power computation.
- Often with specialization, we think of larger changes than just the microarchitecture.
  - Such as with the machine-learning processor
- Other forms of specialized accelerator exist within the SoC.
  - A cryptographic processor (CryptoCell-712) for secure boot, cryptographic functions, etc.
  - A video processor (Mali-V76) for decoding videos
  - A display processor (Mali-D71) for driving displays and offloading some imaging tasks from the GPU

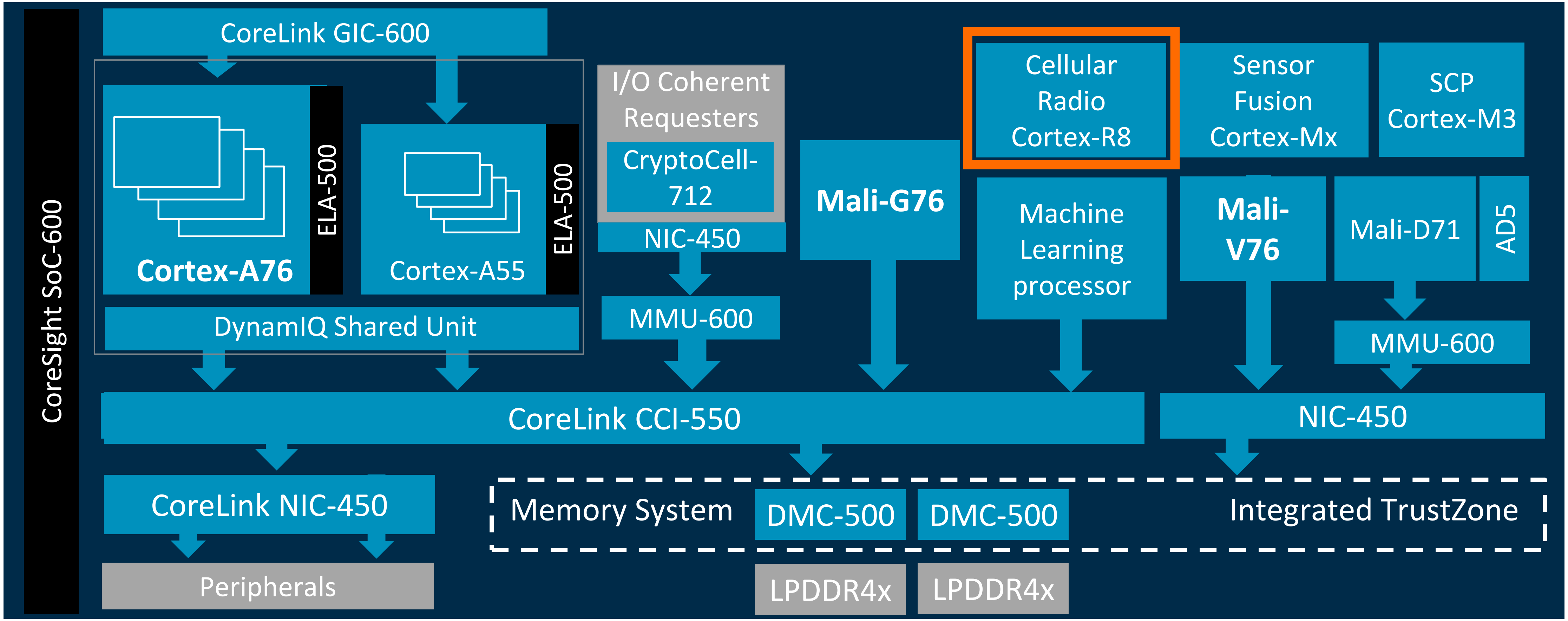
DRAM Interface



Memory Controller (DMC-500)

- What does a memory controller have to do?
  - Convert system memory requests to the necessary series of commands to access the correct rank, bank, row and column in an external SDRAM
  - Buffer and reorder requests to optimize performance and meet QoS goals
  - Error checking and handling
  - Refresh control logic for SDRAM
- What is it connected to? In our SoC example:
  - Each memory controller (Arm DMC-500) supports dual AXI4 (128-bit) system interfaces.
  - Connects to the actual DRAM PHY using a standard interface (called DFI).

Real-time Processing



Real-time Processing

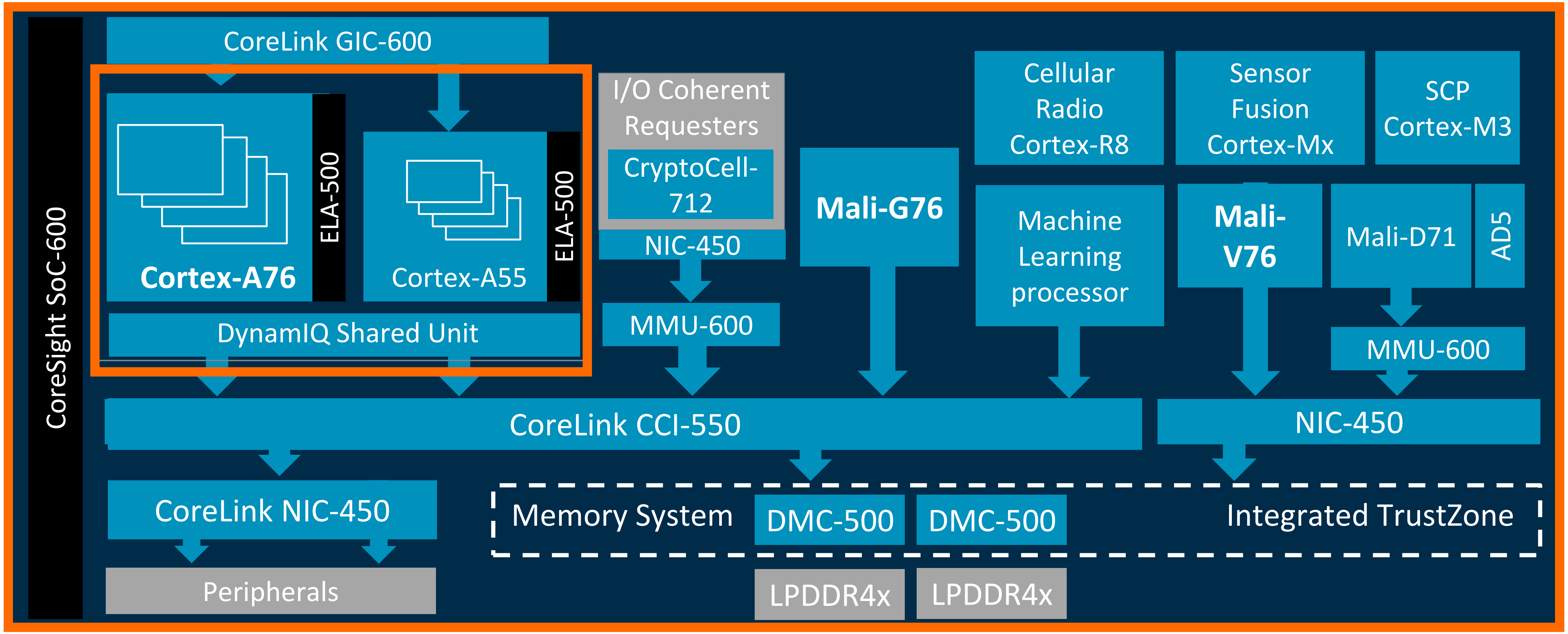
- The software requirements for the latest cellular communications standards are complex.
  - E.g., LTE Advanced Pro and 5G
- Requires real-time multi-core processor
  - In our SoC example, a quad-core coherent cluster of Cortex-R8 cores is used.
- Low-latency and hard real-time requirements
  - Large Tightly Coupled Memories (TCMs) in addition to traditional instruction/data caches
  - Simple Memory Protection Unit (MPU) rather than virtual memory to reduce memory latency
  - Extra interface ports to tightly couple the rest of the latency-sensitive modem system with cores
- Reliability
  - Improved error detection, correction, and containment schemes

Other SoC Design Considerations

SoC Considerations

- Power domains
- Virtualization
- Reliability, Availability, and Serviceability (RAS)
- Safety using Dual-Core Lock Step

Power Domains

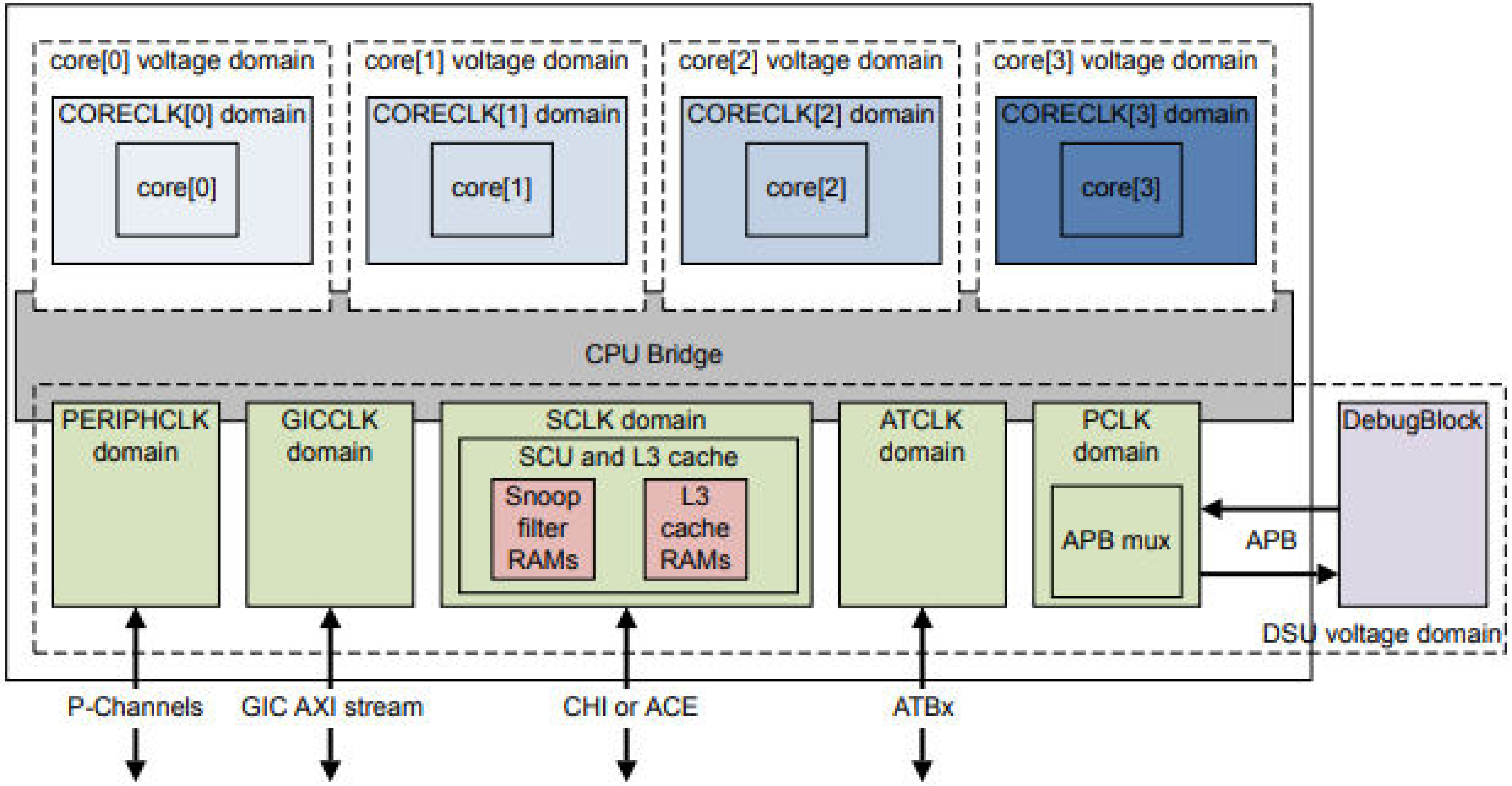


Power Domains

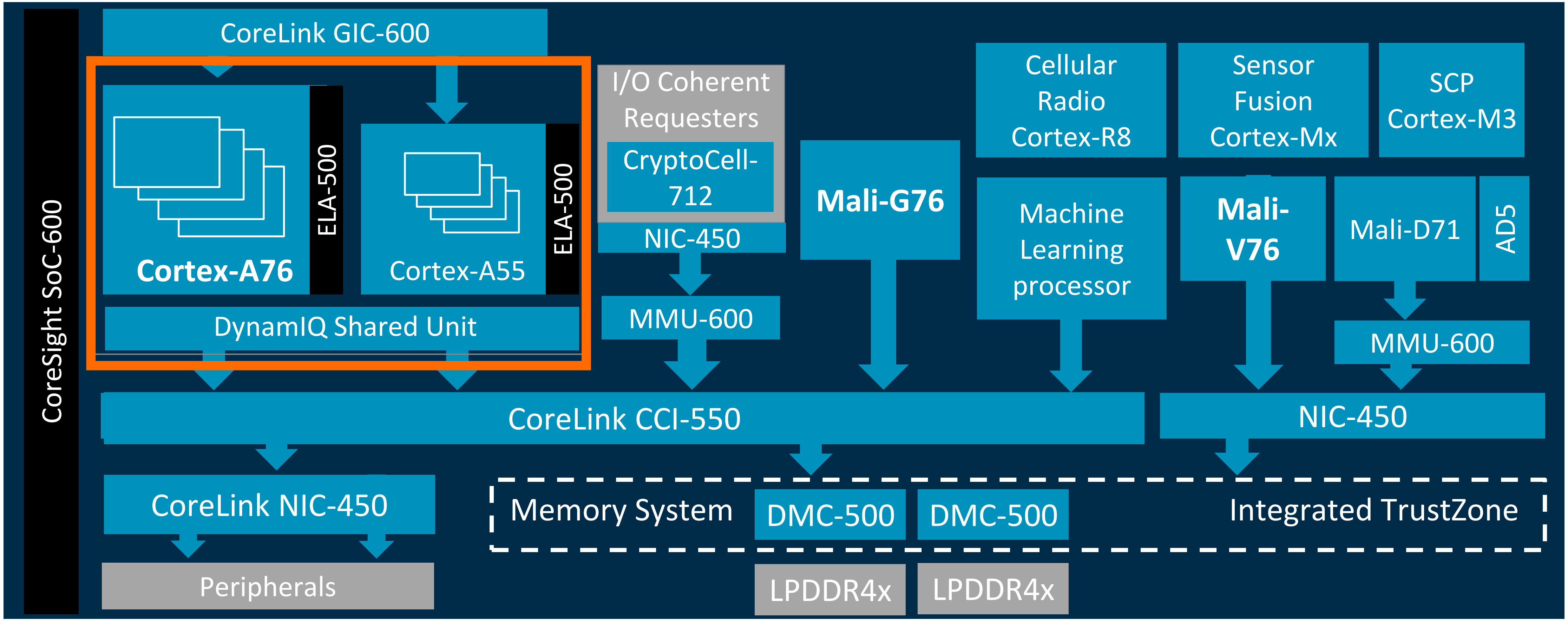
- Providing power to all components of a system-on-chip is challenging.
  - A SoC will implement multiple different domains to create independence between components.
  - This enables power gating when a component is unused.
- There will usually be a system-wide power controller.
  - This can be programmed (through the kernel) to implement a specific power policy.

Power Domains

- Within a DynamIQ cluster alone, there are multiple domains.
  - Each core has its own power domain.
  - Blocks with same color are in the same power domain.



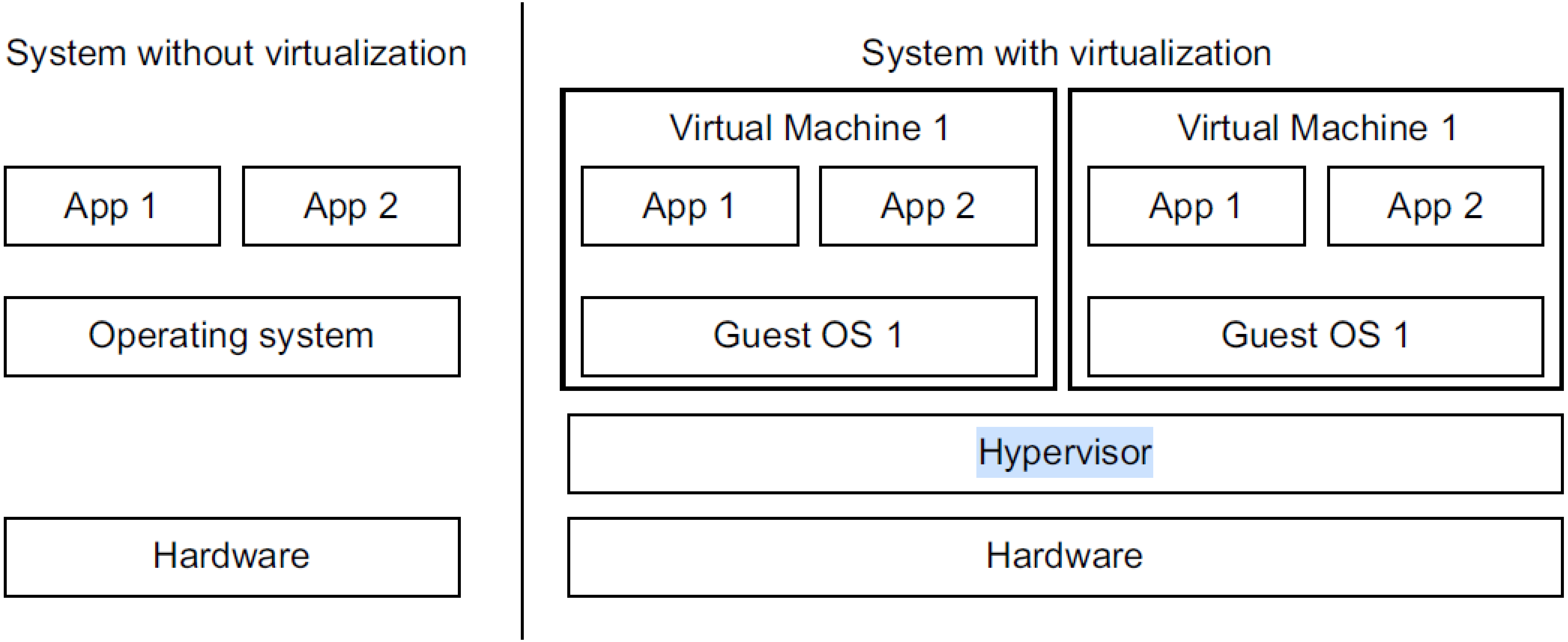
Virtualization



Virtualization

- Virtualization is the ability to create virtual machines that act like real machines.
  - A virtual machine is a software program that mimics a real system called the guest.
    - Being a software program, it actually runs on a real machine called the host.
  - To the user, it is (almost) indistinguishable from a real system.
    - Difficult, if not impossible, to know that they are running on a virtual machine and not real hardware
- A virtual machine contains all the functionality of the real system.
  - E.g., both user and privileged ISAs
  - Peripherals and other devices
  - The ability to run an operating system (the guest)
- The hypervisor provides a virtual system to each of the Guest OS and monitors their execution.

Virtualization



System without virtualization

App 1

App 2

Operating system

Hardware

System with virtualization

Virtual Machine 1

App 1

App 2

Guest OS 1

Virtual Machine 1

App 1

App 2

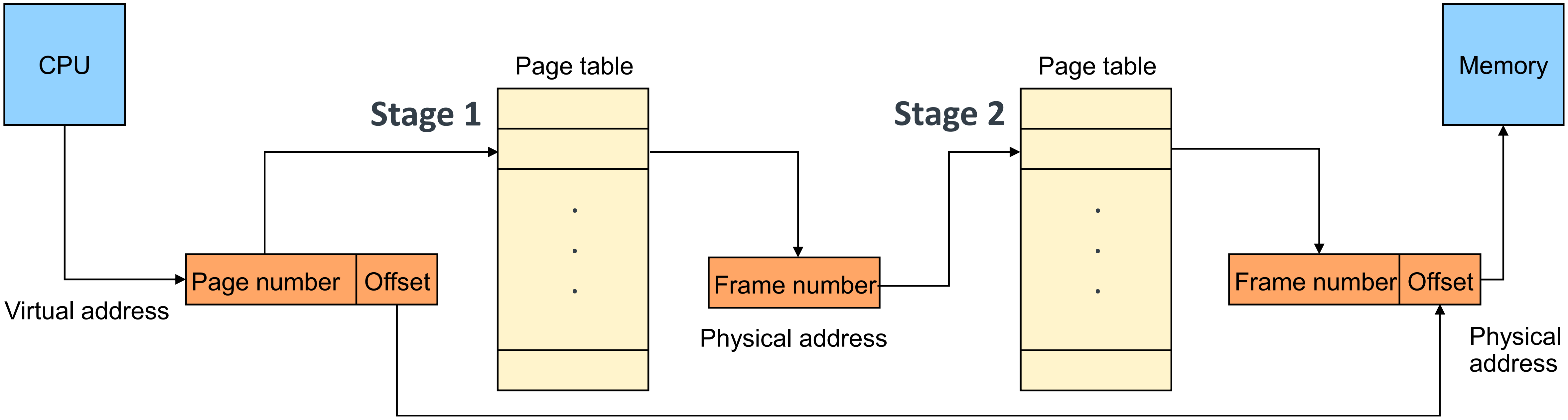
Guest OS 1

Hypervisor

Hardware

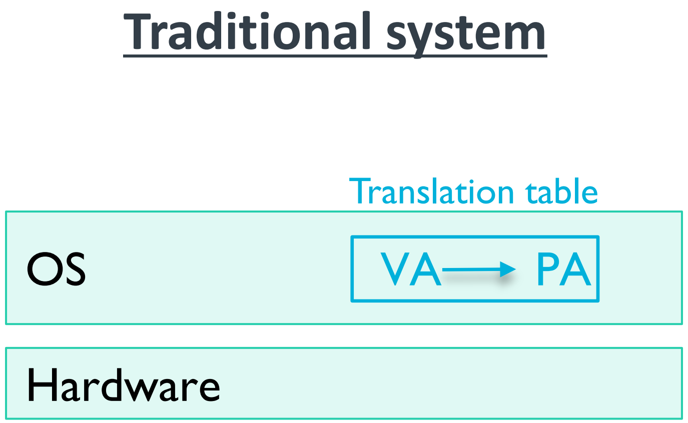
Virtualization

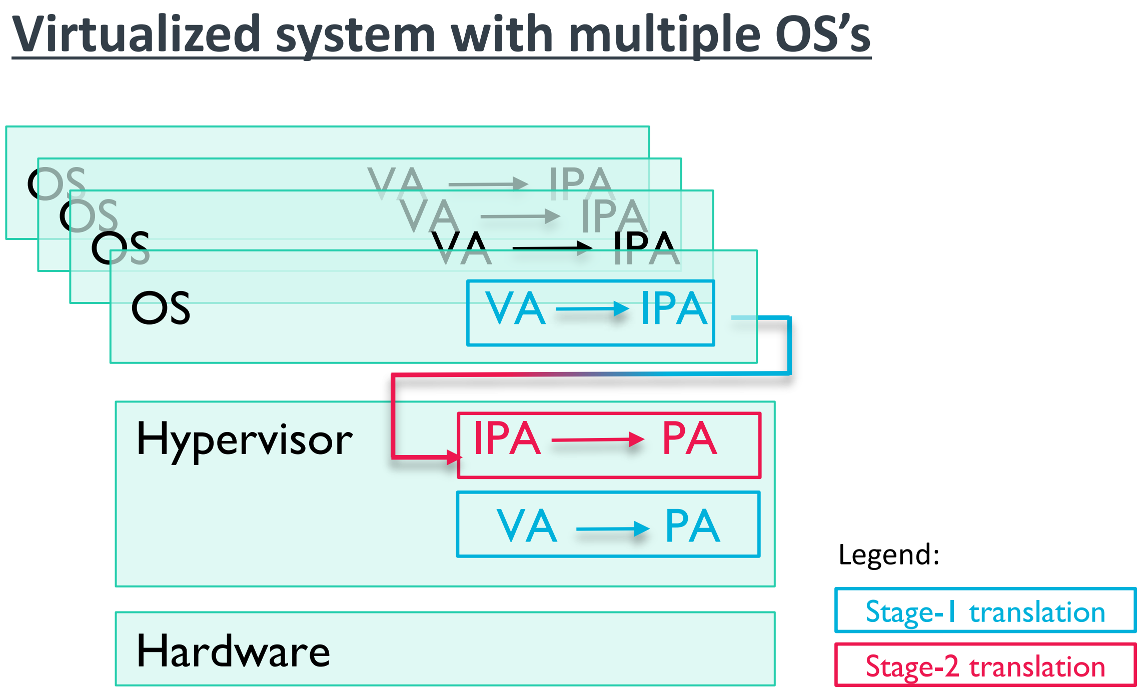
- Hardware support can increase the efficiency of virtualization.
  - Reducing the costs of switching between virtual machines
- E.g., page-table translation for guest OS applications
  - Processor supports a second stage of page-table translation.



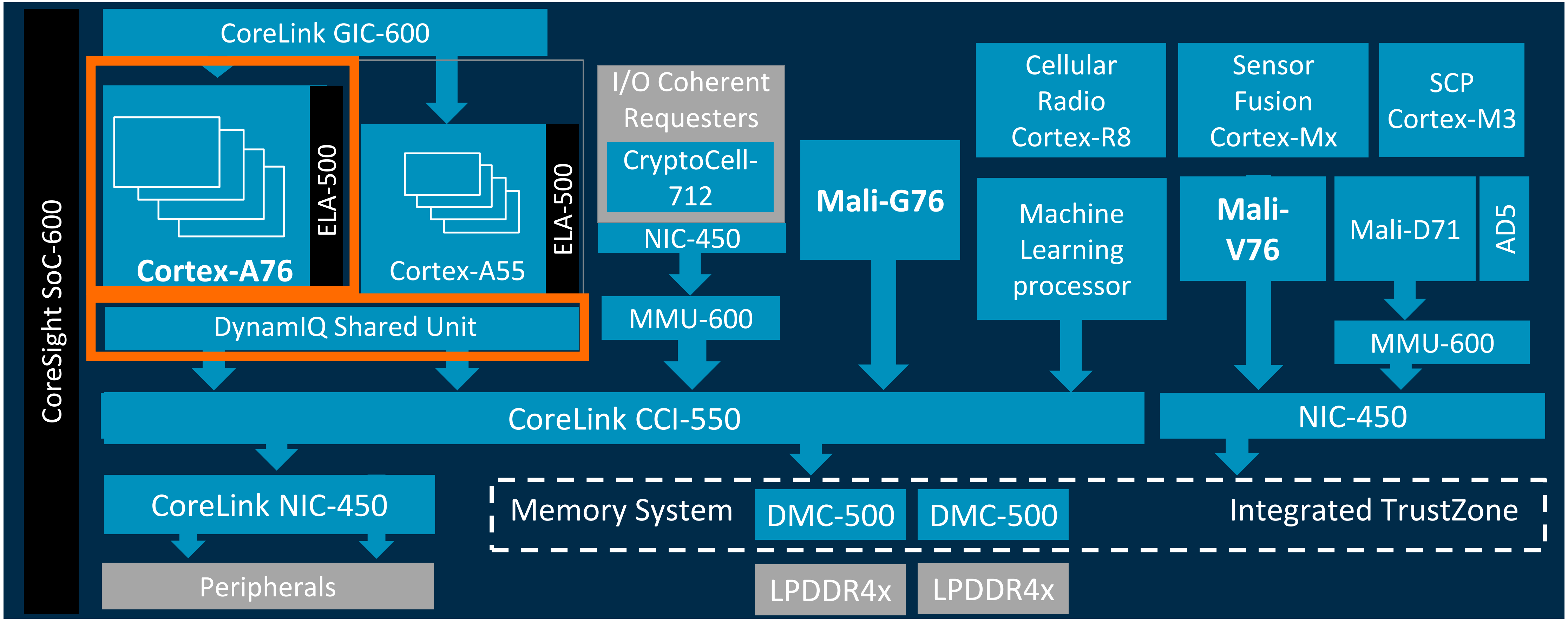
Virtualization

- In virtualization, memory management requires additional translation.
  - The hypervisor has its own translation table.
  - Additional translation stage using Intermediate Physical Address (IPA)



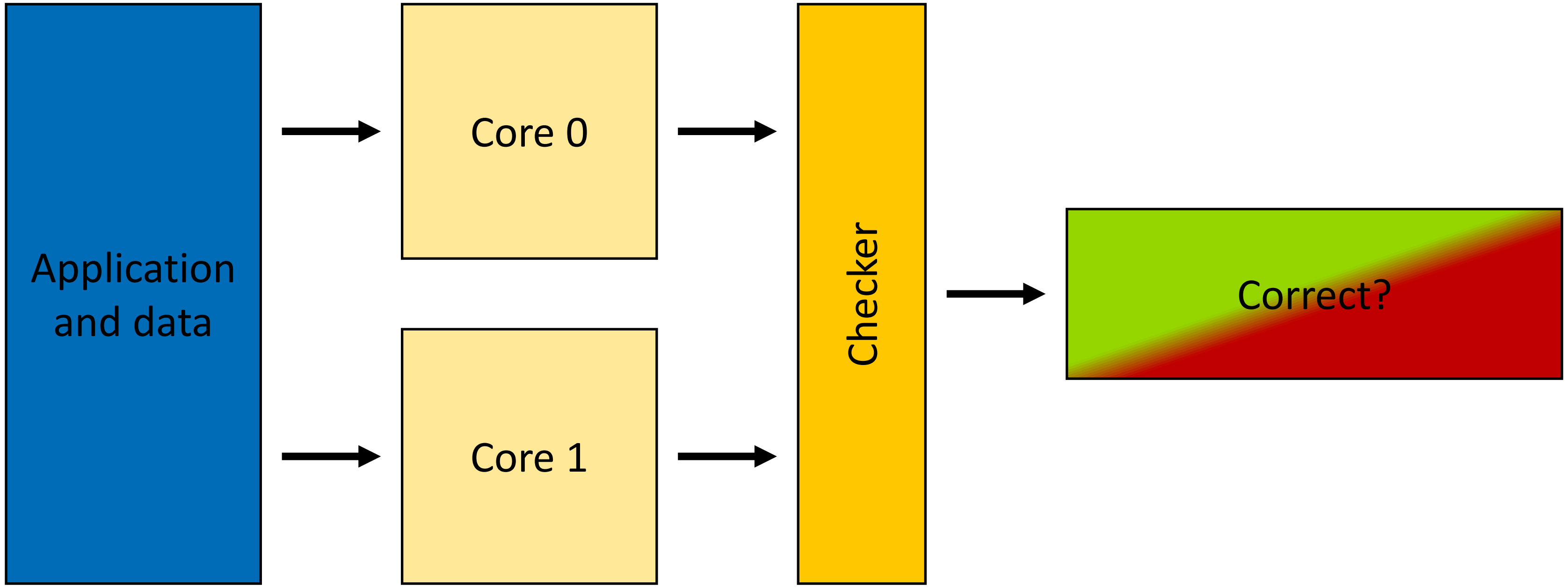


Safety - Dual Core Lock Step



Dual Core Lock Step (DCLS)

- DCLS is an industry-standard technique that runs the program twice on different hardware.
  - Results compared at each cycle
  - Introduces spatial and temporal redundancy into the system



Backup Slides

Heterogeneity in Microarchitectures

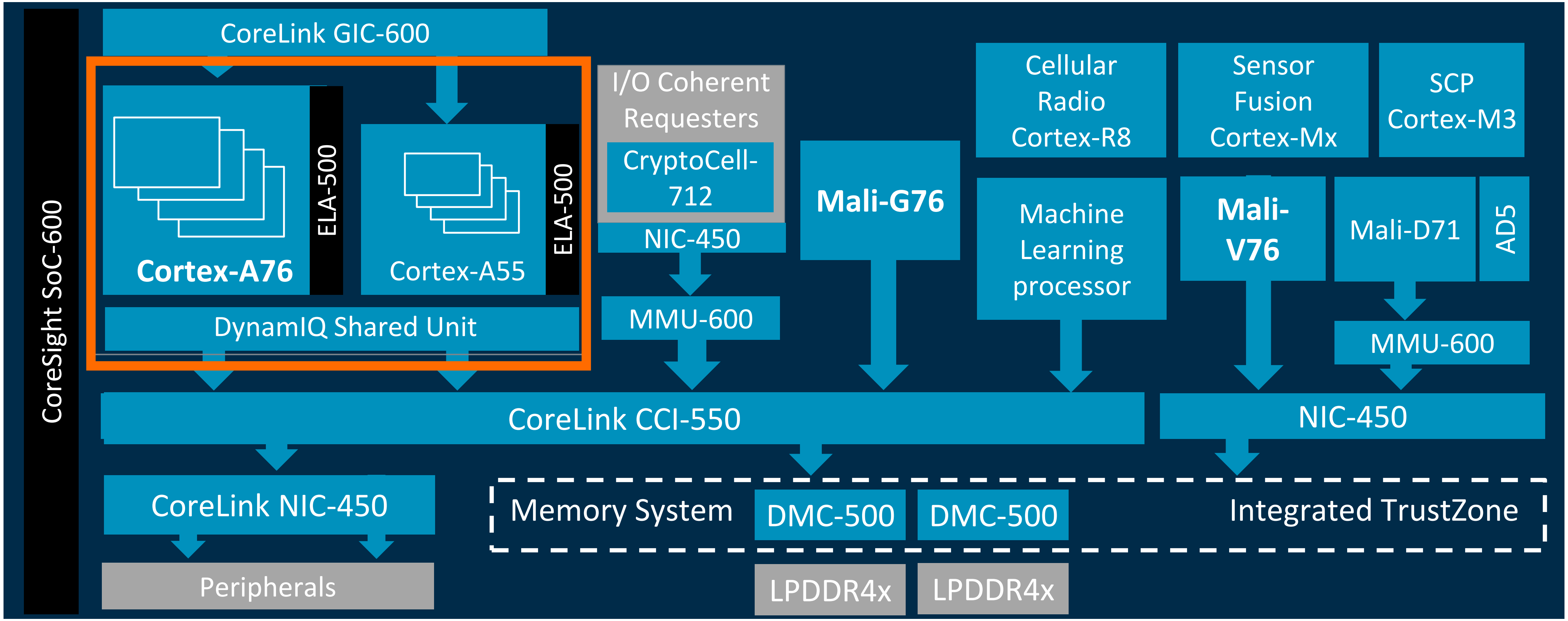
Cortex-A76

- 4-way superscalar, out-of-order processor
  - 8-wide issue
- 13-stage pipeline
- Multilevel branch-target cache
  - Branch unit has 2 × fetch-unit bandwidth
- 128-entry instruction window
- Two load/store pipelines access 64 KB L1D
  - Optimized for memory-level parallelism

Cortex-A55

- 2-wide in-order superscalar
- 8-stage pipeline
  - Sweet spot between power/area and frequency
- Neural network-based branch predictor
  - 256-entry Branch Target Address Cache (BTAC)
- Configurable L1D cache size
  - Fully exclusive of L2
- Independent load & store AGUs
  - Address generation units

Reliability, Availability, and Serviceability (RAS)



Reliability, Availability, and Serviceability (RAS)

- What is RAS?
  - **R**eliability – Continuity of correct service
  - **A**vailability – Readiness for correct service
  - **S**erviceability – Ability to undergo modifications and repairs
- Protects data integrity, makes systems more fault-tolerant
  - Transient errors can be detected and corrected before they cause application or system failure.
  - Failing components can be identified and replaced.
  - Failure can be predicted ahead of time to allow replacement during planned maintenance.

Why RAS?

- Technology scaling provides smaller transistors - but also less reliable.
- Multiple points in a processor's life cycle where hard and soft errors can creep in
  - During manufacture (process variation) or transistor aging
  - As an effect of power fluctuations (voltage droops)
  - Through particle strikes (e.g., caused by cosmic rays, gamma radiation)
- These errors can cause the processor to perform calculations incorrectly.
  - Either because a circuit has been corrupted (hard error)
  - Or bits in the values used have been flipped (soft error)
- Memory systems are most vulnerable.
  - Especially DRAM, which contains small transistors, easily flipped
  - But also caches, due to having a lot of memory cells close together, increasing the probability of a particle hit

RAS Techniques

- Terminology
  - An *error* is any deviation of the correct behavior.
  - A *fault* leads to an *error* – example of fault: manufacturing problem.
- Techniques for Reliability
  - Avoid error, correct the error, contain the scope of the error
- Techniques for Availability
  - Fault handling
  - Minimize scope of failure
- Techniques for Serviceability
  - Fault and error reporting

RAS in Armv8-A Profile

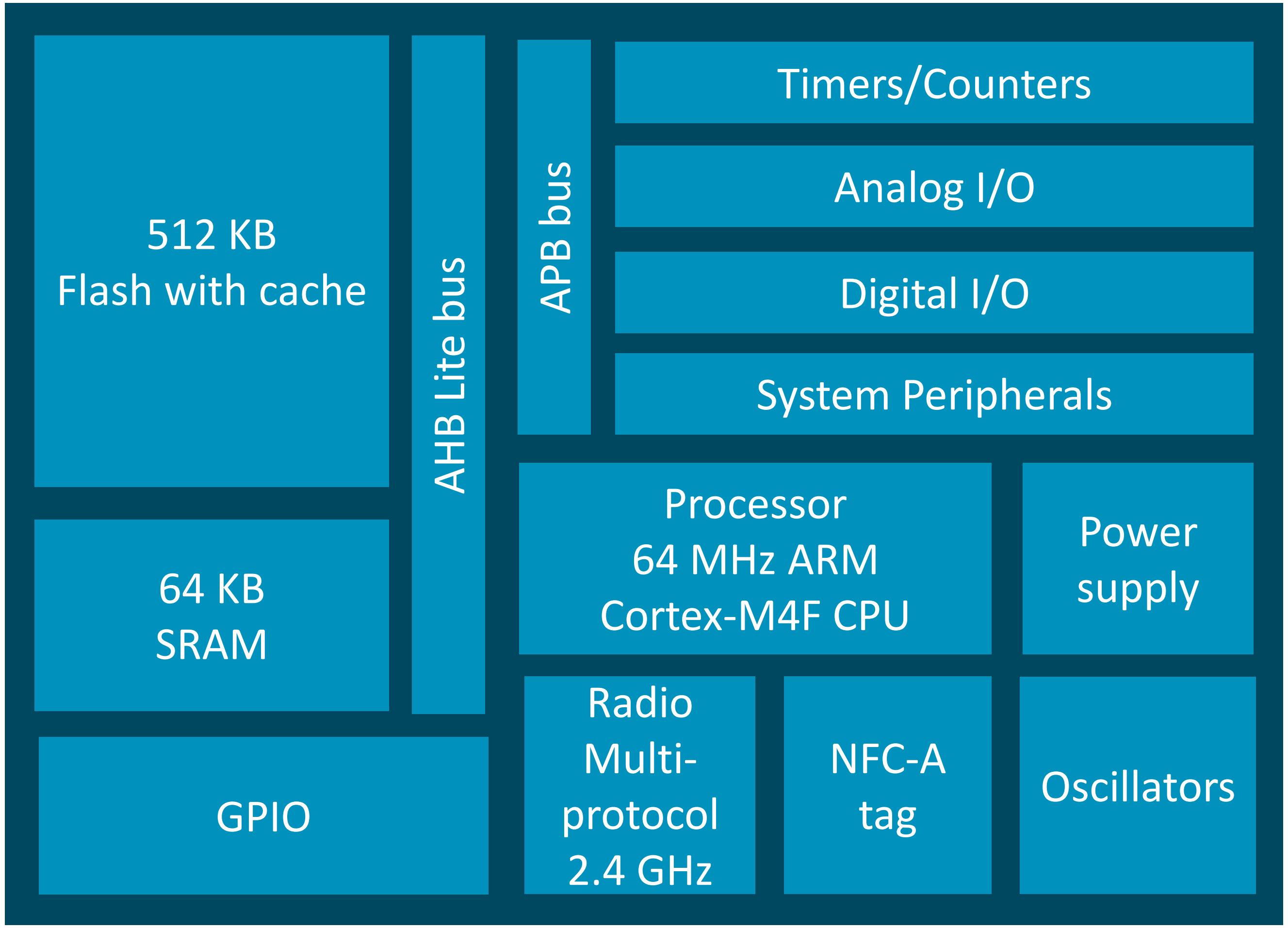
- Arm introduced RAS specification for the Armv8-A profile.
- Includes:
  - Error recording in specific RAS registers.
  - Supports fault injection for the purpose of testing fault handling software by programming some Error Record and Control registers.
  - Optional cache protection with SED, interleaved parity, and SECDED capabilities.
  - Error recovery and fault handling interrupt outputs.
  - Error Synchronization Barrier instruction (`ESB`), which allows efficient isolation of errors

Other SoC Examples

Further Examples: Wireless IoT Soc

Nordic Semi nRF52832

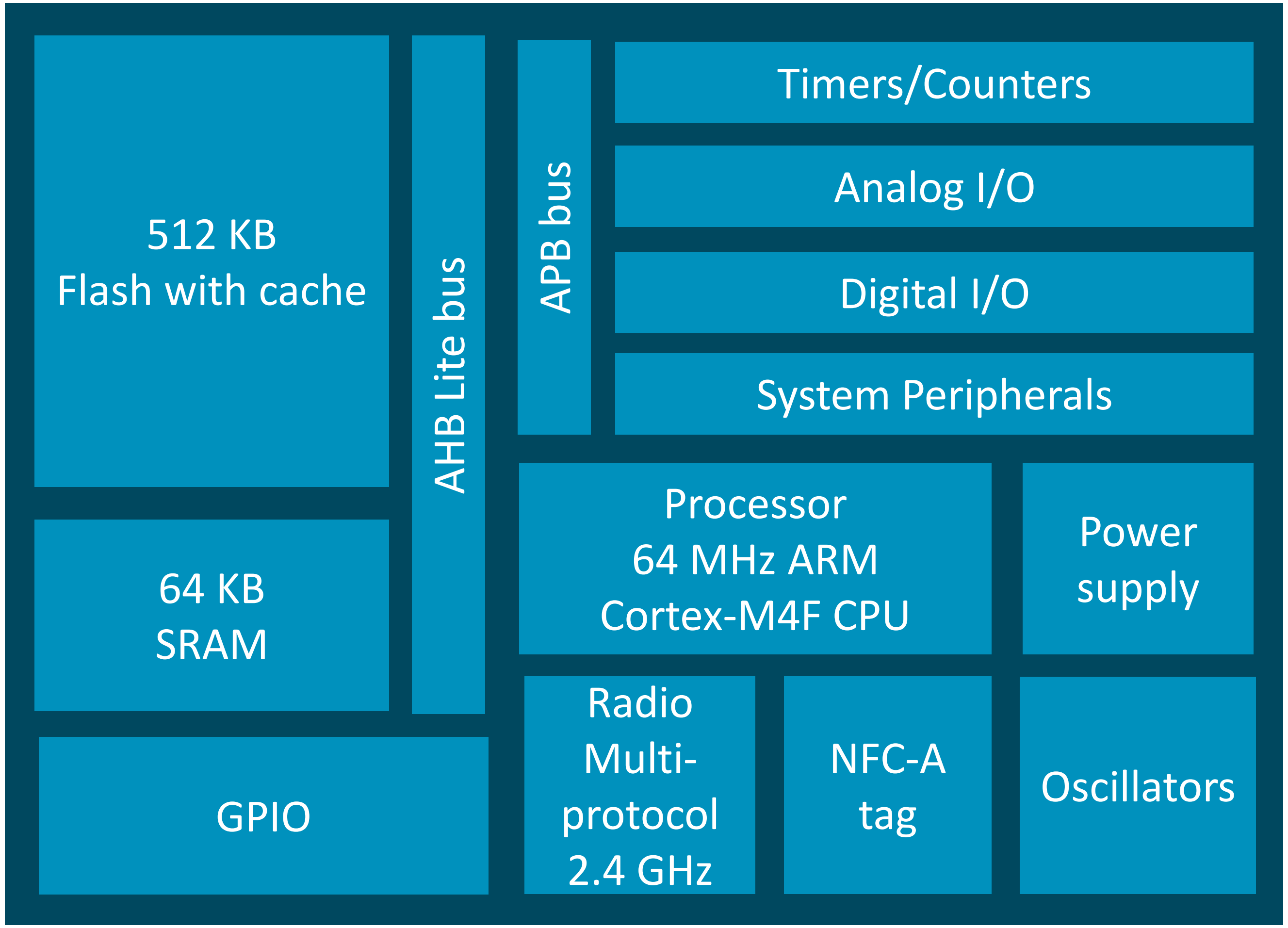
- Bluetooth 5 Low Energy SoC for smart home, sensor networks, wristwatches, remote control applications, etc.
- Multiprotocol 2.4 GHz Radio
- NFC-A Near Field Communication
- 64 MHz ARM Cortex-M4F
  - With floating point ("F)" and DSP instructions
  - 3-stage pipeline with branch speculation
  - Various sleep modes
- Cost $2.50 or less at volume



Further Examples: Wireless IoT Soc

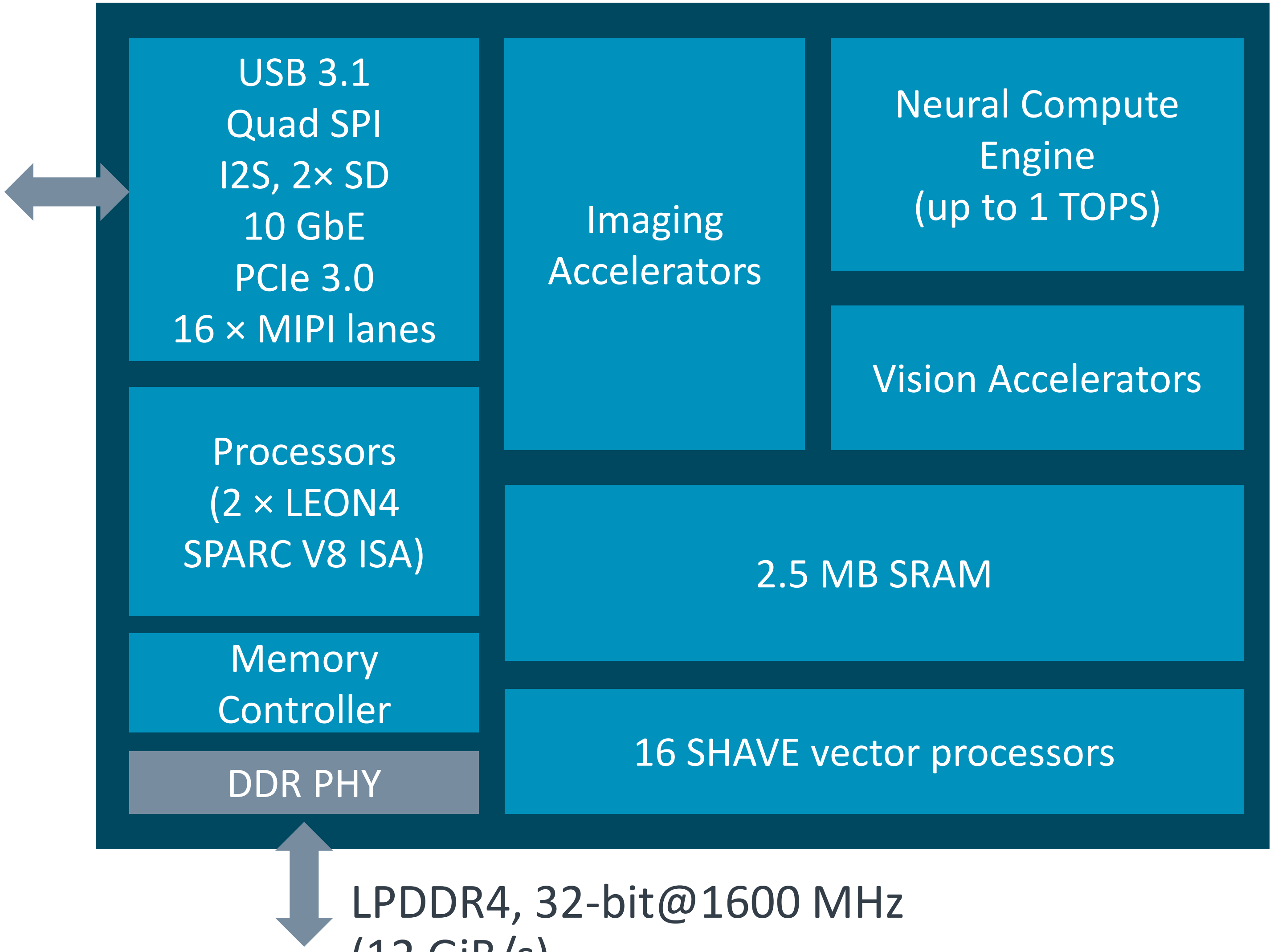
Nordic Semi nRF52832

- 32 pin GPIO
- 12-bit/200 kSPS ADC
- SPI/2-wire/I^{2}S/UART/PDM/QDEC
- 128-bit AES ECB/CCM/AAR co-processor
- On-chip DC/DC buck converter
- Ultra-low standby current (0.3-1.6 µA @3 V), up to 15.2 mA when transmitting
- 3 mm × 3.2 mm WLCSP50 package (or 6 mm × 6 mm QFN48)



Further Examples: Intel Movidius Myriad X

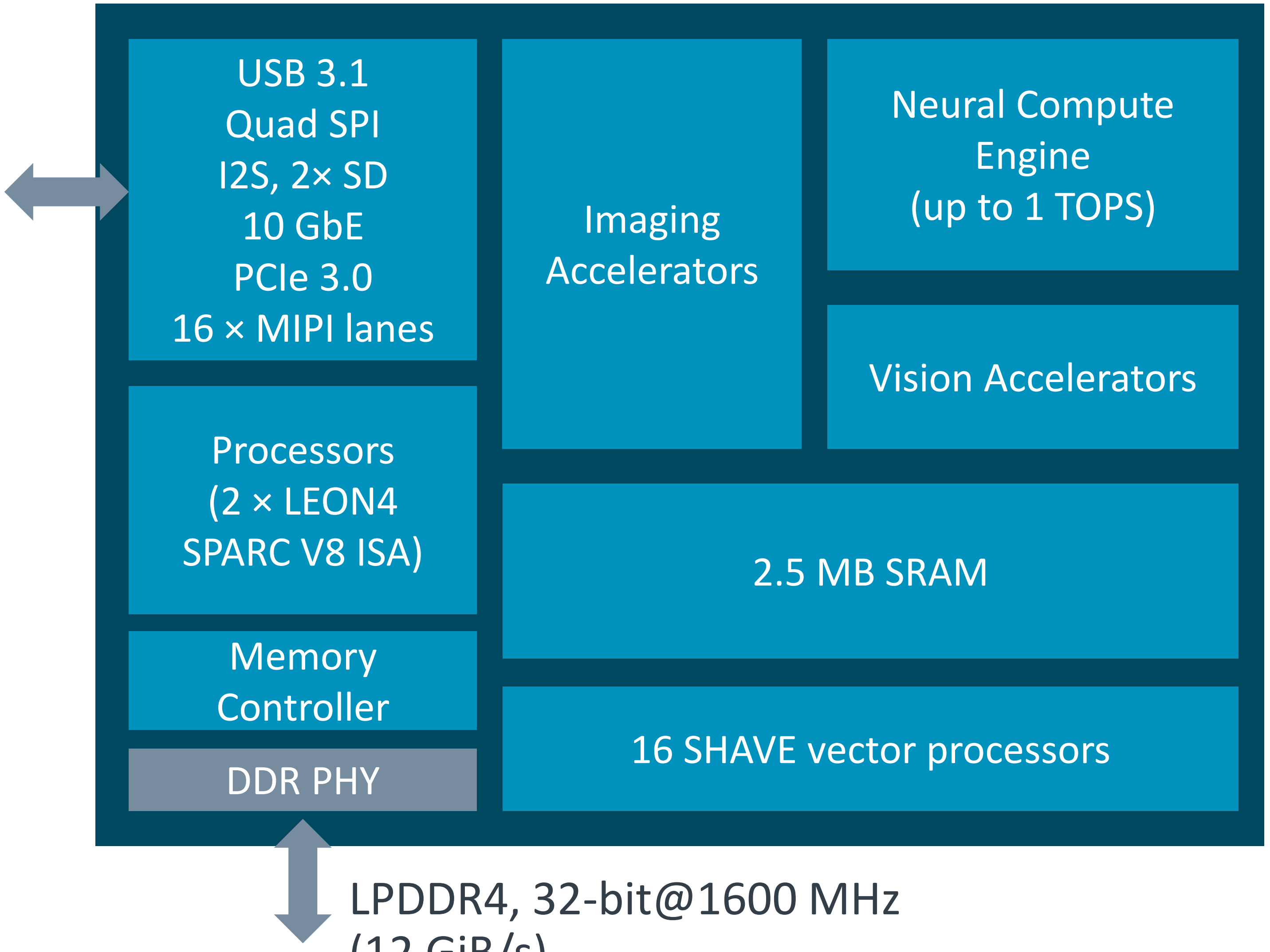
- Vision Processing Unit (VPU)
- Intelligent machine vision, e.g., robotics, augmented and virtual reality, wearables, and smart city applications
- >4 TOPS
- Low-power <1 W
- package 8.1 mm × 8.8 mm
- Process: TSMC's 16 nm FinFET



LPDDR4, 32-bit@1600 MHz
(12 GiB/s)

Further Examples: Intel Movidius Myriad X

- 20+ fixed-function imaging and vision accelerators, 4K video HW encoder, neural compute engine (DNN accelerator)
- **SHAVE vector processors**
- VLIW-style processor with 128-bit vector operations. Each processor can perform multiple scalar and vector operations in parallel.



LPDDR4, 32-bit@1600 MHz
(12 GiB/s)