OpenAMP: "**Open A**symmetric **M**ulti-**P**rocessing" Project

Runtime coexistence and collaboration

Runtime hardware resource assignment

Resource sharing and IPC between runtimes

Control mechanisms to start and stop runtimes

Typical system: Linux + RTOS on one system-on-chip

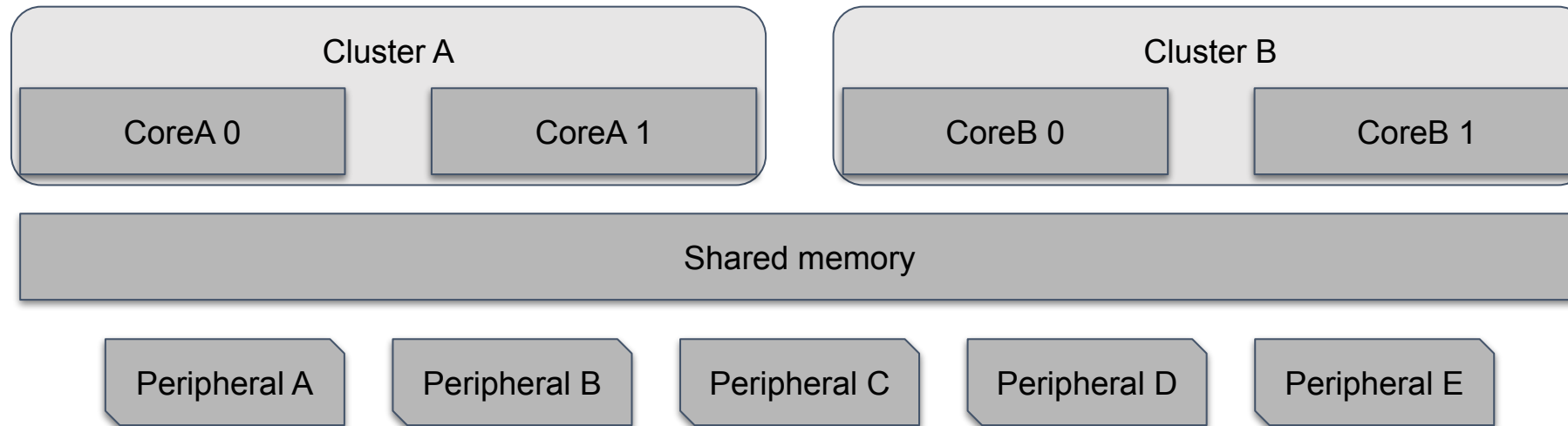www.openampproject.org.

OpenAMP Project Intro
Standardizing Asymmetric Runtime Integration

OpenAMP

OpenAMP Embedded Targets

Modern Embedded Targets integrate multiple HW resources e.g. multiple core clusters, shared memory and peripherals

Cluster A		Cluster B	
CoreA 0	CoreA 1	CoreB 0	CoreB 1

Shared memory

Peripheral A	Peripheral B	Peripheral C	Peripheral D	Peripheral E

OpenAMP Embedded Runtimes

Embedded Targets have multiple Runtimes that need to collaborate

Linux + Apps	RTOS App	Bare Metal App

OpenAMP HW Assignment

The HW resources need to be assigned into Runtime Domains

| Linux + Apps | RTOS App | Bare Metal App | Linux Domain |

| CoreA 0 | CoreA 1 | CoreB 0 | CoreB 1 | RTOS Domain |

| Memory Linux | Memory Shared | Memory RTOS | Memory Shared | Memory Bare Metal | BM Domain |

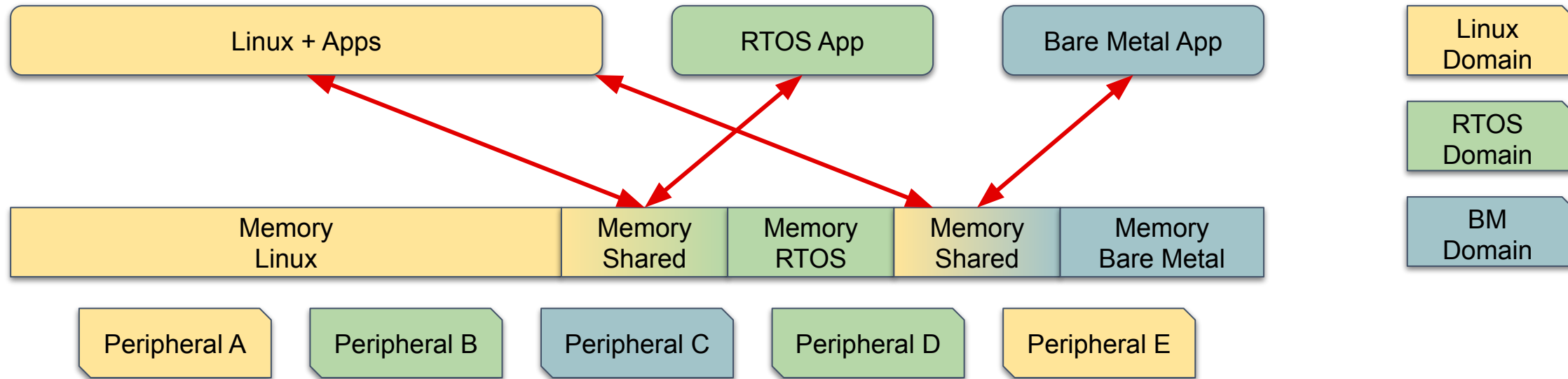| Peripheral A | Peripheral B | Peripheral C | Peripheral D | Peripheral E |
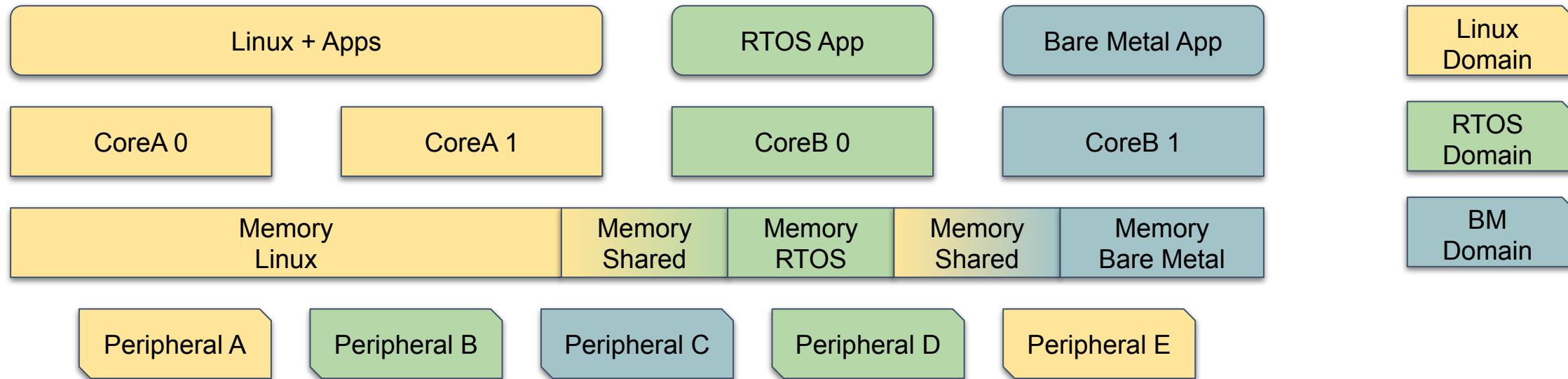
OpenAMP Runtime Control

**The Runtimes need to be managed,
e.g. loaded into memory and started**
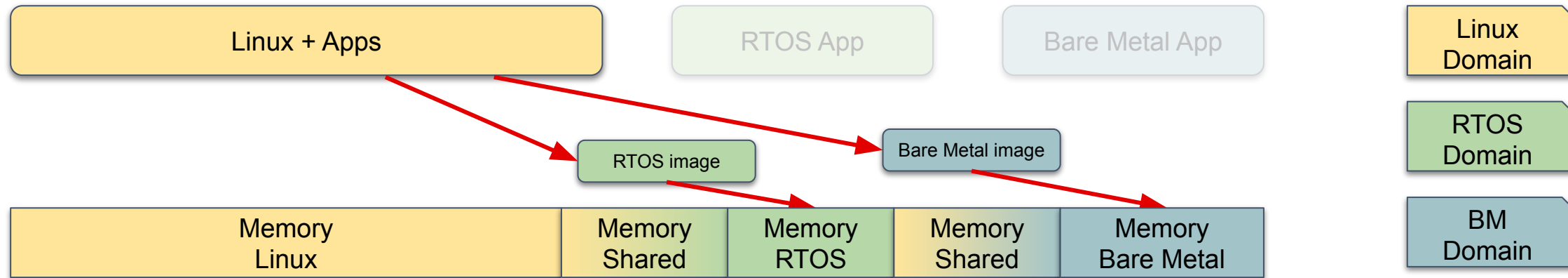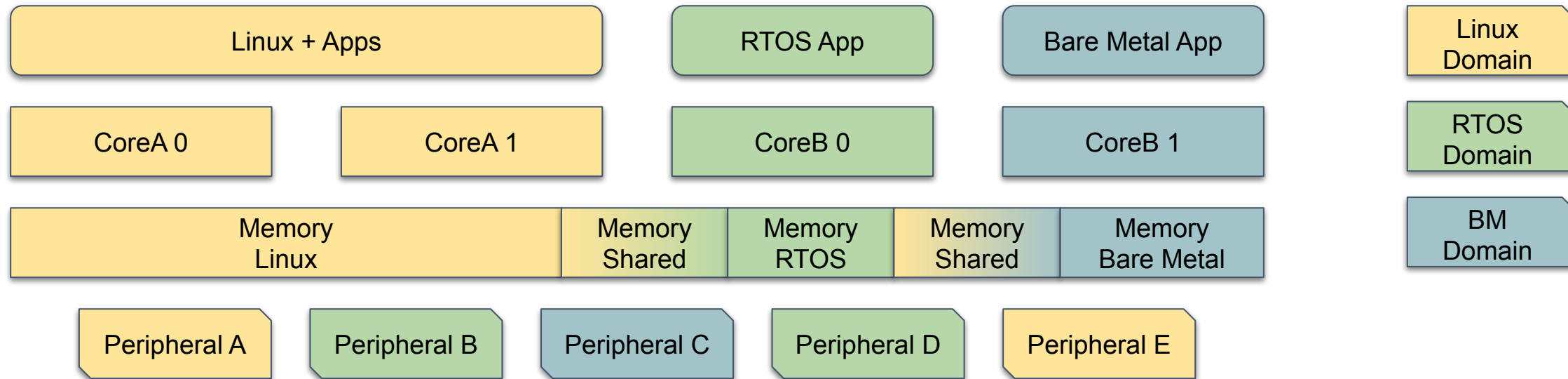
OpenAMP Resource Sharing and IPC

The Runtimes need to share data, services, and virtual devices

OpenAMP Mission

OpenAMP provides standards, runtime libraries and tooling built on top of existing open source projects to simplify runtime collaboration

Linux + Apps	RTOS App	Bare Metal App	Linux Domain

CoreA 0	CoreA 1	CoreB 0	CoreB 1	RTOS Domain

Memory Linux	Memory Shared	Memory RTOS	Memory Shared	Memory Bare Metal	BM Domain

| Peripheral A	Peripheral B	Peripheral C	Peripheral D	Peripheral E

OpenAMP Technologies

- [Remoteproc](#)
  - A Linux subsystem for loading and controlling coprocessors
- [RPMsg](#)
  - A simple IPC message system with multiple ports and name server
- [Virtio-msg](#)
  - A virtio transport that can be used in AMP systems (and more)
  - Leverage existing virtio protocols and drivers:
    - -net, -blk, -vsock, -console, -fs
    - -i2c, -gpio, -spi
- [System Devicetree](#)
  - Extension of Devicetree to express a whole AMP system
  - Used for coordinated configuration and partitioning of the system
  - [Lopper](#): a tool set for System Devicetree

- Other technologies that align with the mission can be added over time

HW Example: AMP SoC
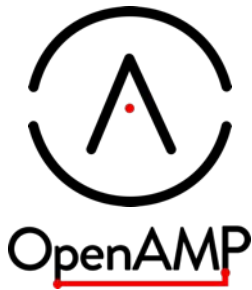
- A single SoC
- CPUs that are SMP Linux capable
- Other CPUs are MCU like, used for
  - Real time or IO offload
  - Safety or Security critical functions
  - Digital Signal Processing
  - Low Power standby w/ IO
- Examples:
- ZynqMP: 4x A53s + 2x R5s [+FPGA]
- STM32MP15: 2x A7s   + 1 M4
- NXP iMX8M+: 4x A53s + 1 M7 + DSP
- TI TDA4VM: 2x A72s + 6x R5s + 3 DSPs
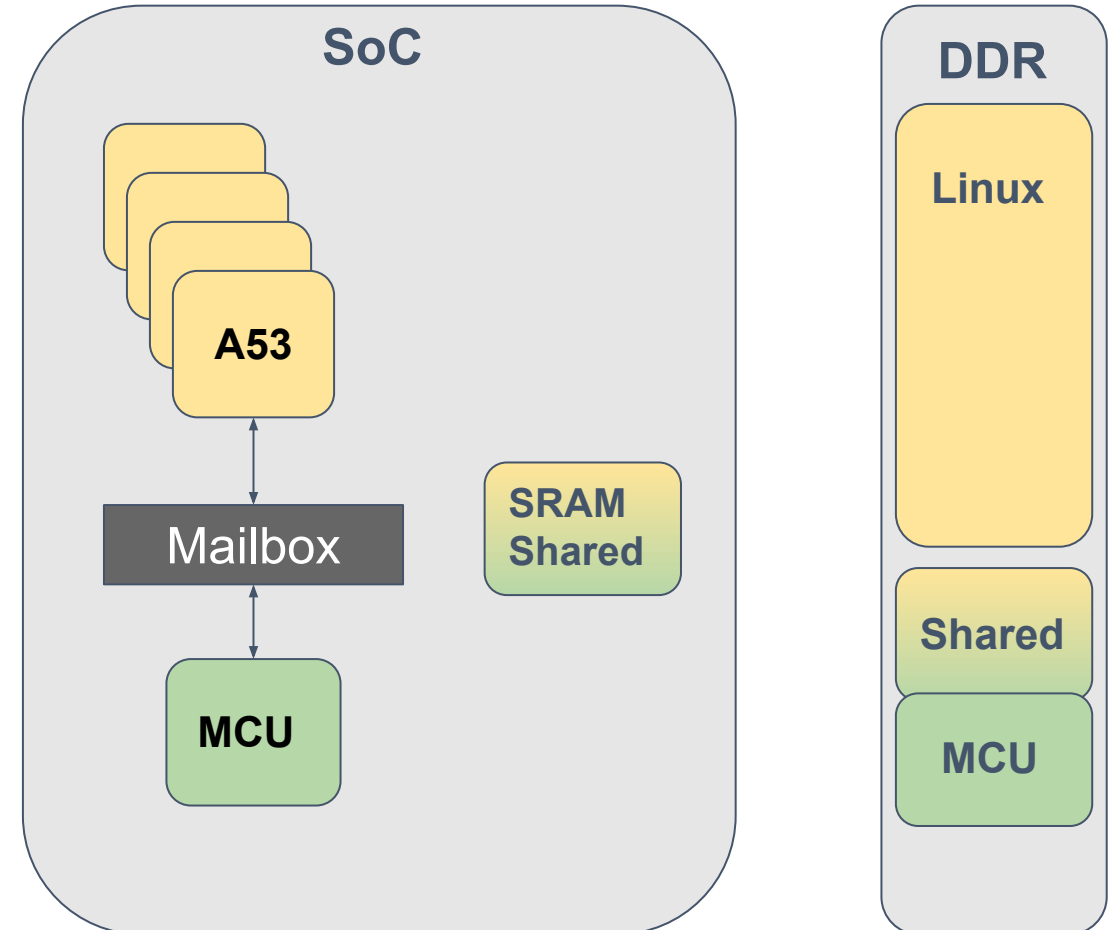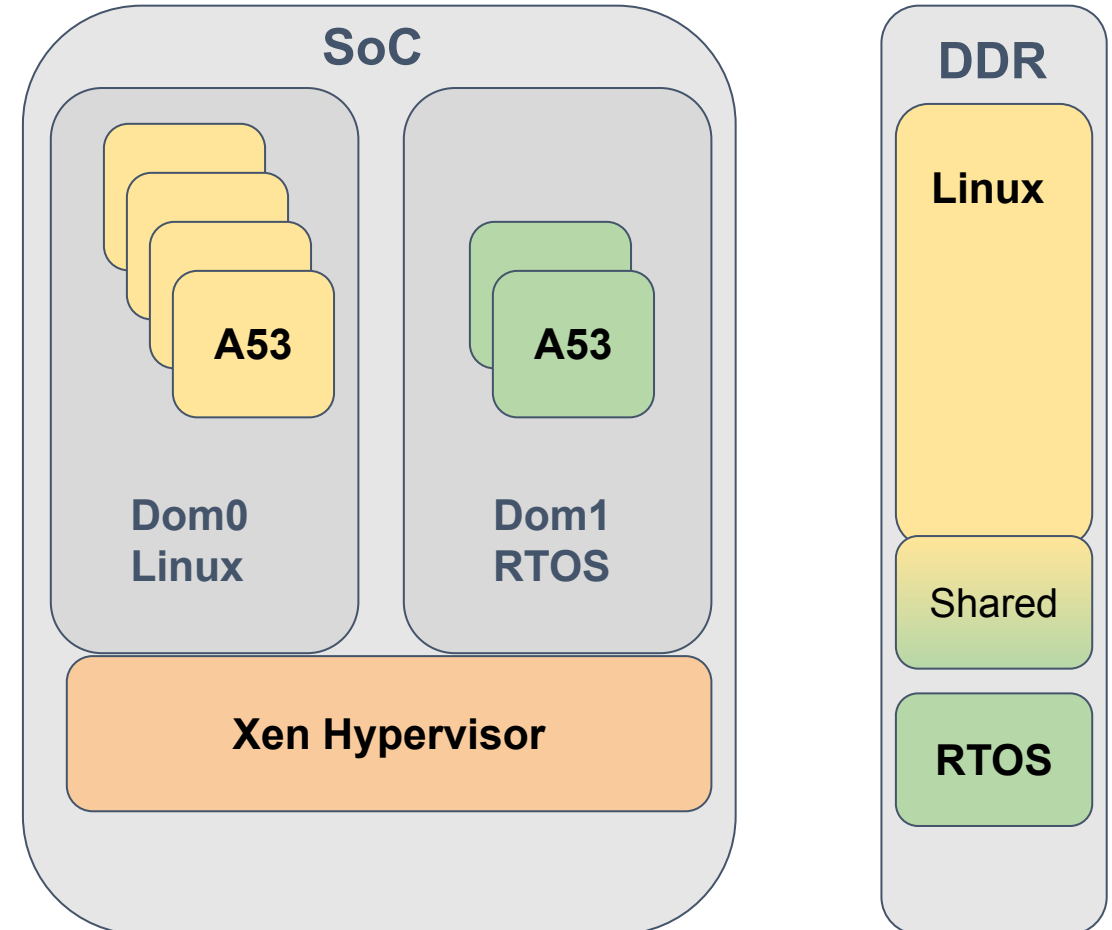
HW Example: AMP via PCIe (and similar)
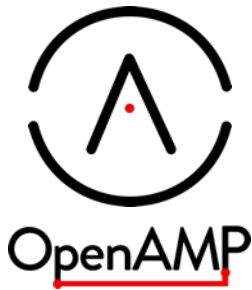
- x86 host with Arm SoC on a PCIe card
- Two PCIe RC systems connected with a non-transparent bridge
- UCIe and Chiplet ecosystem
  - Making these AMP systems more common and more customizable
- Two QEMUs using IVSHMEM
  - Good stand-in for the cases above
  - Approximation of a non-transparent bridge
  - Shared memory, MSI interrupts, and a doorbell MMR on each side
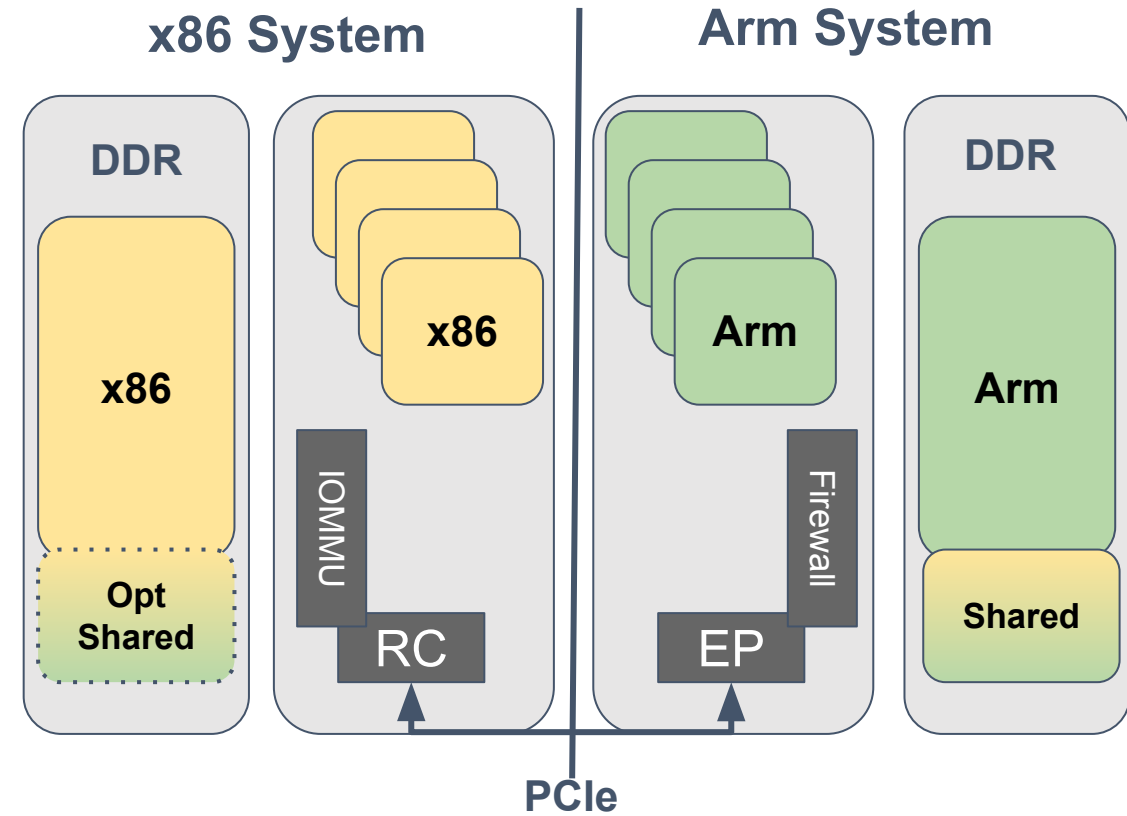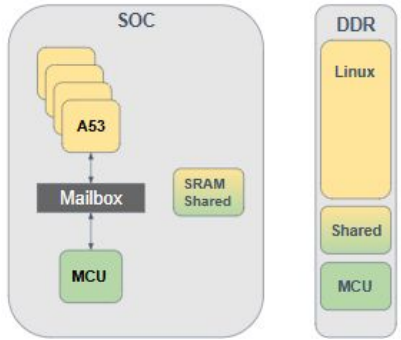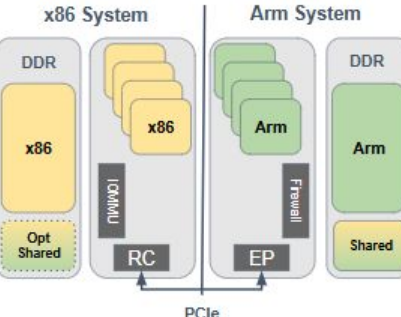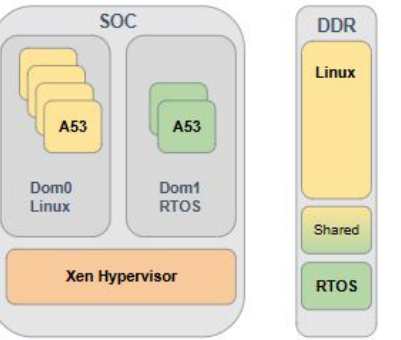
HW Example: Mixed Critical system w/ hypervisor

- Example hypervisor: Xen
- Dom0less creation of critical RTOS domains at boot time
  - Real-time
  - Higher level of Functional safety
- Linux based Dom0 boots in parallel
- Other physical CPUs and memory can be used by Dom0 to create non-critical DomUs.

SW Examples

AMP SoC:
- Linux on Cortex-A, Zephyr on Cortex-M/R
- Devices & memory assigned w/ System Devicetree using Lopper
- Start Zephyr through remoteproc
- Communicate through rpmsg
- Zephyr exposes portion of I^2C bus via virtio-msg based virtio-i2c

AMP via PCIe (and similar):
- Each system boots independently
- Arm system access the x86 rootfs using virtio-fs over virtio-msg
- x86 uses a the Arm as a smart-nic using virtio-net over virtio-msg

Mixed critical system with Xen hypervisor:
- Xen has multiple ways to do paravirtual devices, but OpenAMP virtio-msg allows:
  - low complexity for RTOS implementation
  - very low vCPU interference for Real-time and Functional Safety
  - minimal hypervisor requirements
- Same model can apply to any hypervisor
- Devices, CPUs & memory partitioned w/ System Devicetree using Lopper

Check it out and get involved!

Community Project Website

www.openampproject.org

Member companies:

Thank You