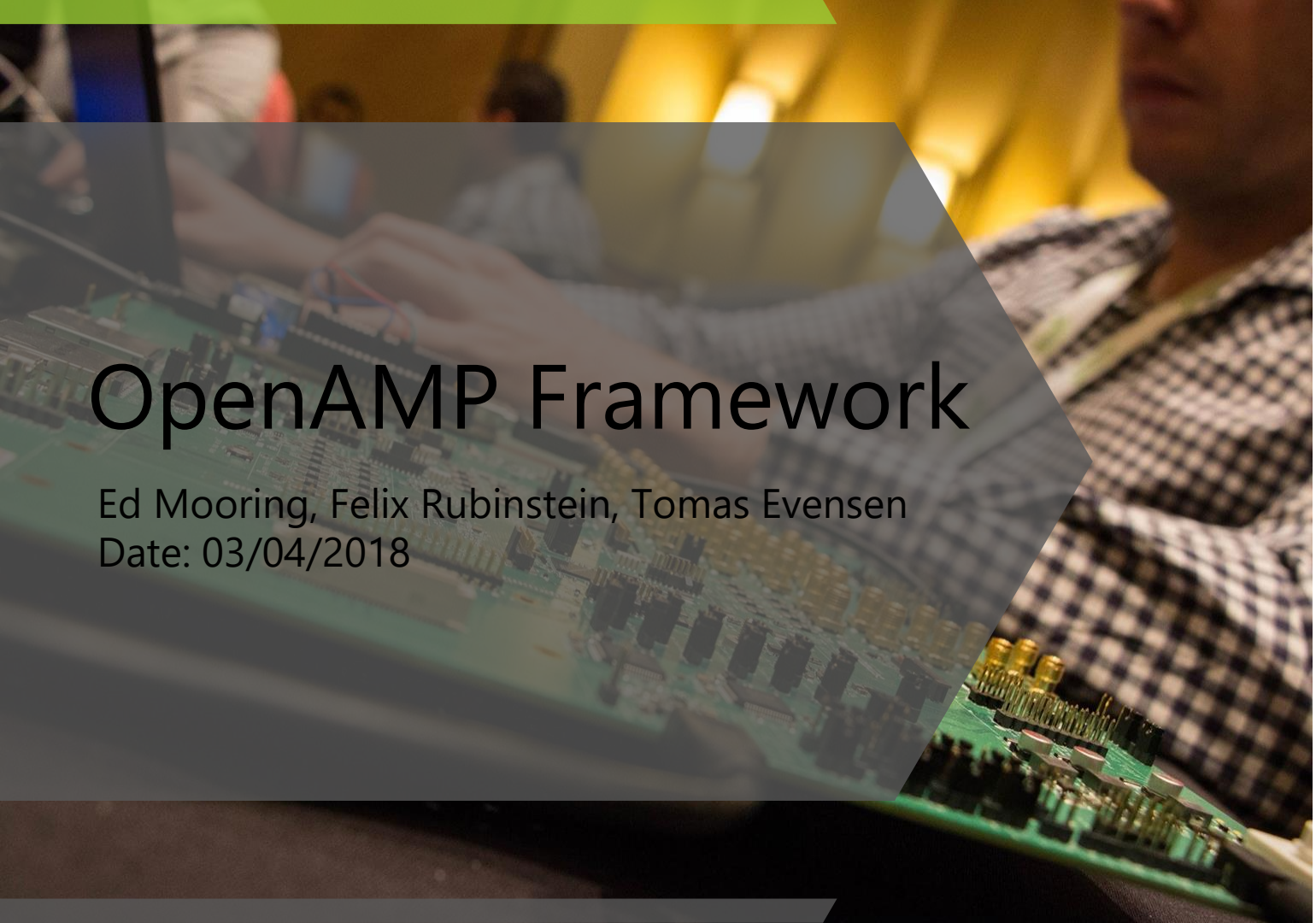
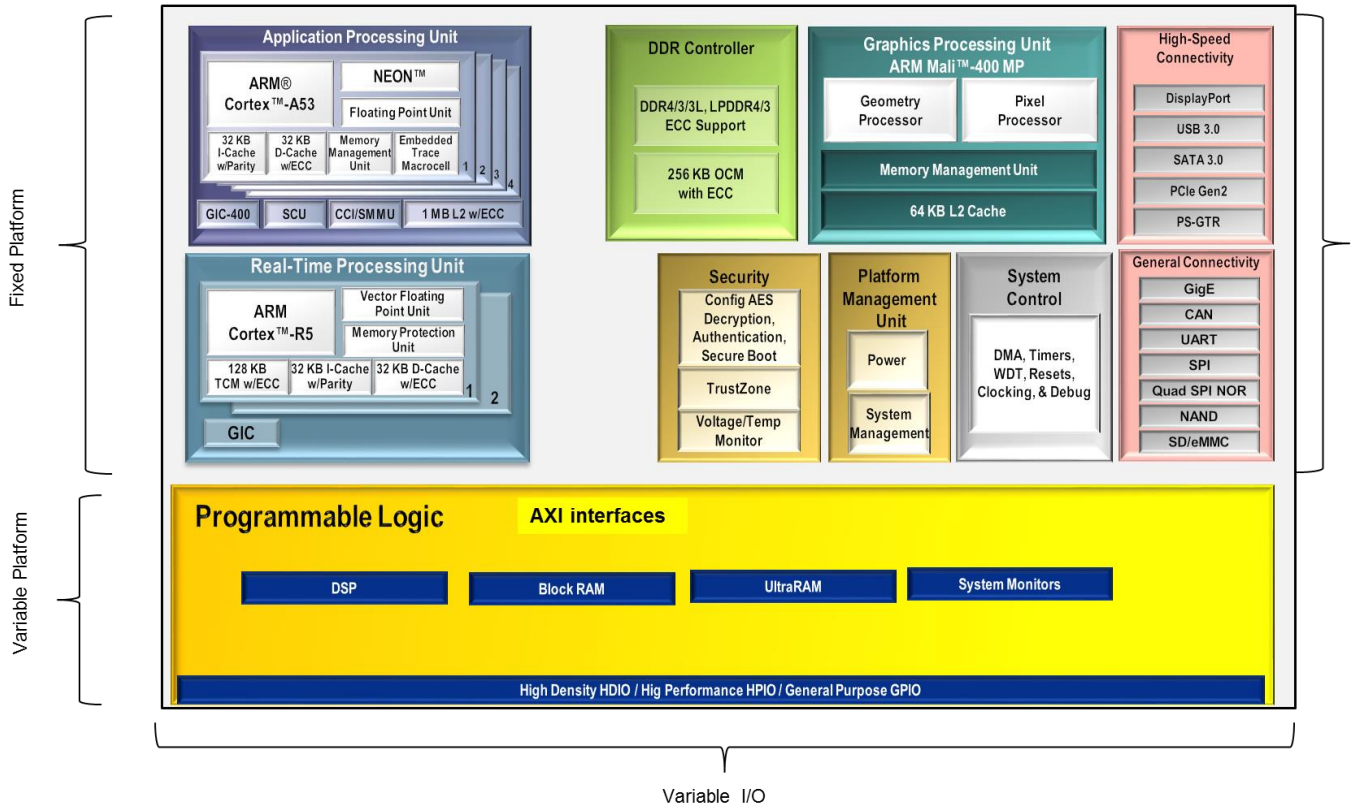


OpenAMP Framework

Ed Mooring, Felix Rubinstein, Tomas Evensen
Date: 03/04/2018

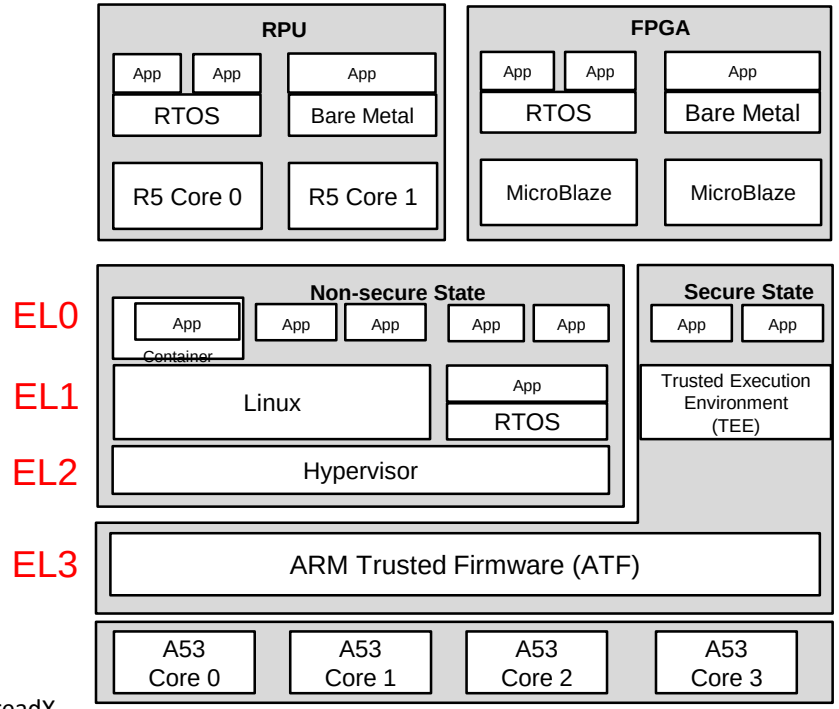
Linaro connect
Bangkok 2019

Zynq® UltraScale+™ MPSoC



Execution Environments within Zync US+ MPSoC

- Multiple core clusters
  - A53, R5, MicroBlaze
- Multiple Execution Levels (EL)
  - EL0 – User space – Linux apps, Containers, RTOS apps
  - EL1 – OS space – Linux kernel, RTOS + RTOS apps
  - EL2 – Hypervisor – Xen, Jailhouse, …
  - EL3 – Firmware – ARM Trusted Firmware
- Multiple Security Environments
  - TrustZone (TZ) – HW protecting resources (e.g. memory)
  - Trusted Execution Environment (TEE) – SEL1
- Multiple Operating systems
  - Linux (including Android) is used in majority of use cases
  - Most free and commercial RTOS's are being used
    - FreeRTOS, Zephyr, VxWorks, Integrity, Nucleus, uC/OS, OSE, ThreadX
    - QNX/Neutrino, Sciopta, eT-kernel, Lynx, PikeOS, …
  - Bare metal (no OS) is common on smaller cores
  - OS often pinned to specific core for embedded applications



Can We Simplify SW for Heterogenous Environments?

- Today, most heterogeneous environments are clobbered together ad-hoc
  - Everybody coming up with their own shared memory scheme
- Can we standardize how environments interact?
  - How to configure the environments?
  - How to manage (lifecycle) the environments?
  - How to pass messages between environments?
  - How to share resources between environments?
  - How to port any OS on top of a standardized abstraction layer?
- Can we have an open source implementation solving these problems?
  - Based on already existing open source projects?

These are the questions OpenAMP tries to answer

Executive Summary

- OpenAMP is an open-source framework to interact with heterogeneous SoCs
  - Facilitates use of processing resources for complex designs
- Standardization effort and open-source project
- Evolving AMP/OpenAMP Roll-out
  - From foundation to advanced capabilities
    - APU as master
    - RPU as master
    - Authentication, Decryption of executables
    - Multiple memory types and coherency, zero copy, etc.
  - Arbitrary executable management
  - OpenAMP executable management

Glossary

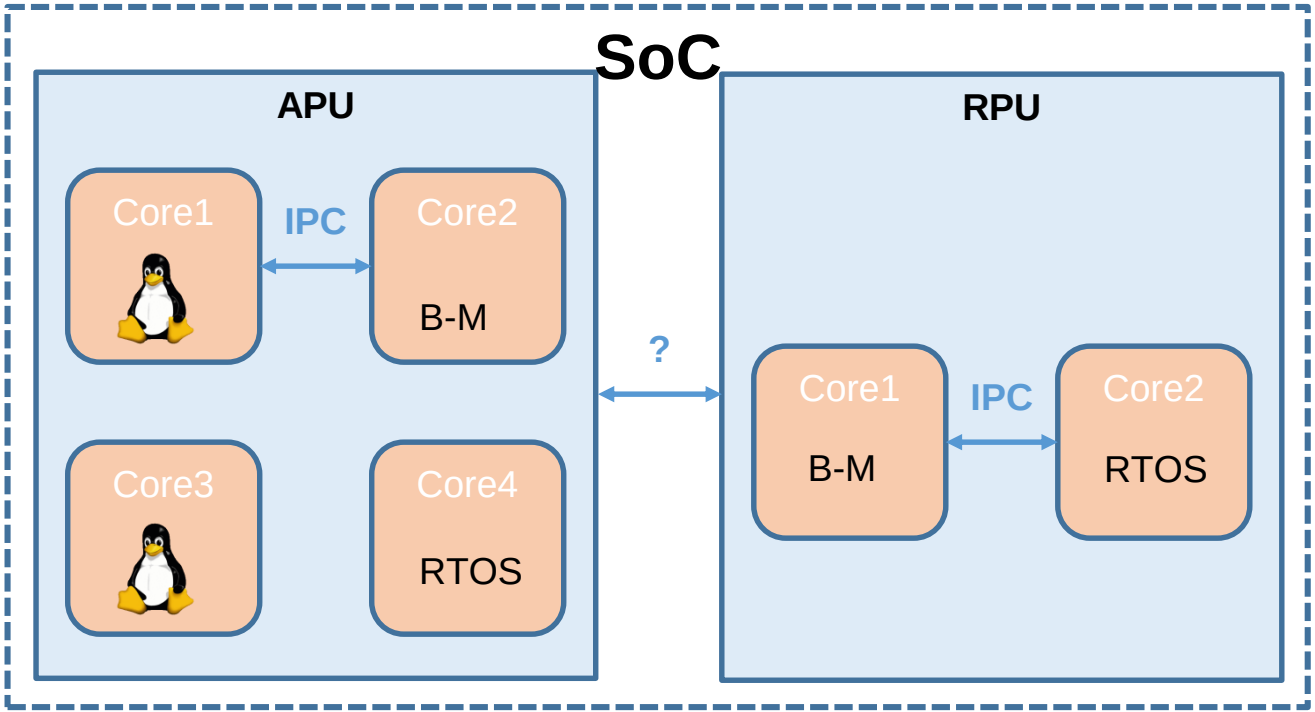
- **SMP: Symmetric Multi-Processing**
- **AMP: Asymmetric Multi-Processing**
- **APU: Application Processor Unit**
- **RPU: Realtime Processor Unit**
- **LCM: Life Cycle Management**
- **IPI: Inter-Processor Interrupt**
- **IPC: Inter Process Communication**
- **HSA: Heterogeneous Software Architecture**

Heterogeneous Software Architecture forced by ZynqUS+ MPSoC

- Not possible to run Linux across Cortex-A and Cortex-R
- AMP implied by differing PUs: APU and RPU
- GPU still abstracted through Libraries/API
- APU a good candidate for Linux
- RPU a good candidate for an RTOS
- Heterogeneous OSes are also needed for homogenous cores
- This can be solved either with unsupervised AMP or a hypervisor

AMP on Heterogeneous SoC

- Interface between APU and RPU will be device-specific
- Abstraction becomes more complicated
- Openly documented framework that different vendors could leverage to abstract device-specific interfaces would be ideal…



Background: Linux AMP

- Heterogeneous SoCs are by no means a “new” concept
  - But we’re seeing more that give developers access to “raw” firmware and that deploy multiple ARM architectures

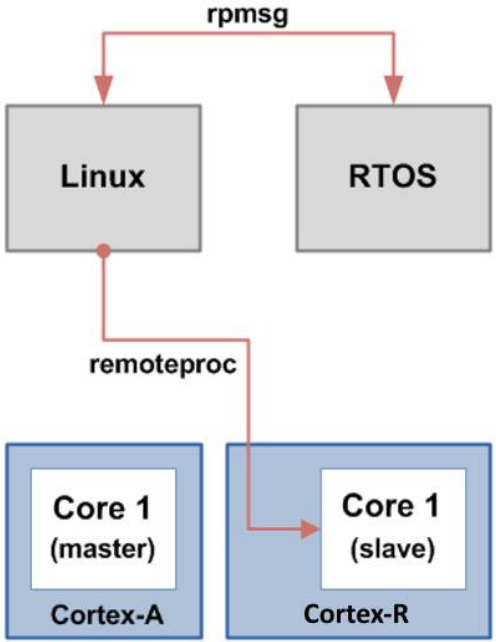
- A Linux framework called ***rpmsg*** and ***remoteproc*** is proof of this
  - Introduced into the Linux kernel around 2011

- **remoteproc – Remote Processor**
  - Framework that allows a Linux master to control/manage remote processors (power on/off, reset, load firmware)

- **rpmsg – Remote Processor Messaging**
  - Messaging framework that provides inter-processor communication (IPC) between kernel drivers and remote processors

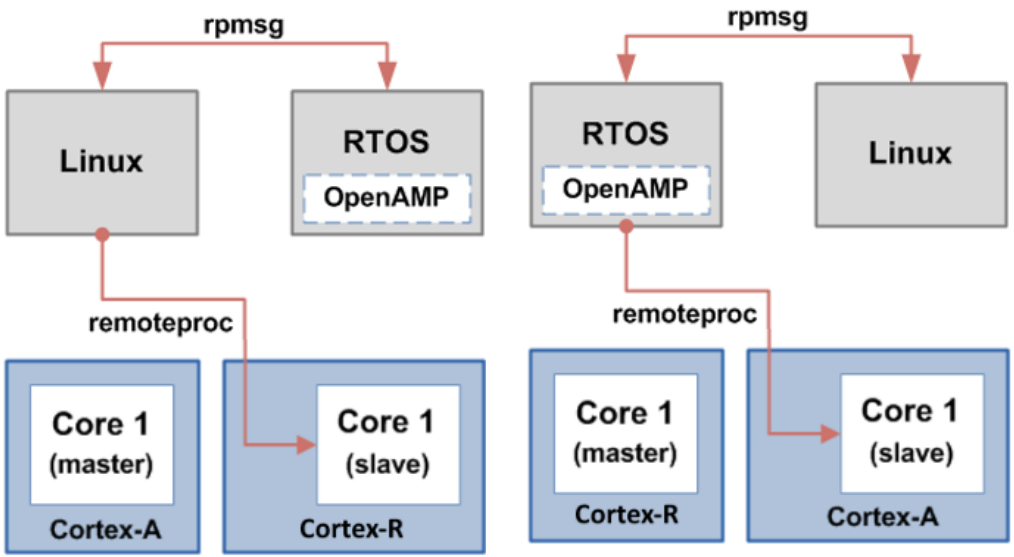
Linux Remoteproc

- The AMP framework was introduced to Linux due to an increasing number of heterogeneous hardware platforms
  - Introduced in 2011
  - Available as of Linux 3.4.1
- The key components of the framework are based on two responsibilities
  - **Management**
    - The **remoteproc** component is a mechanism that allows the Linux master to start software on a remote processor
  - **Messaging**
    - The **rpmsg** component is remote processor messaging that provides inter-processor communication (IPC)
- Linux AMP framework was limited in scope
  - Masters expected to be Linux
  - No framework provided for firmware on remote processors



OpenAMP Framework

- The OpenAMP framework was introduced to expand the scope of the original Linux AMP framework
  - Provides a software framework for remote processors (for example, RTOS or bare-metal)
  - Adopts the same conventions as with Linux (remoteproc and rpmsg)
  - Master no longer needs to be Linux-based
- Introduced by Mentor Graphics in collaboration with Xilinx in 2014
- Clean-room implementation (BSD license)



What is OpenAMP?

- OpenAMP standardizes how Operating Systems interact:
  - Between Linux and RTOS/bare-metal
  - In multicore heterogenous systems
- Includes:
  - Lifecycle APIs to start/stop/restart other OSes (RemoteProc)
  - Inter-Process Communication APIs to share data (RPMsg)
  - Shared memory protocol for OS interactions (VirtIO)
- Guiding principles
  - open-source implementations for Linux and RTOSes
  - Prototype and prove in open-source before standardizing
  - Business friendly APIs and implementations to allow proprietary solutions

OpenAMP libraries

- Lifecycle Management (LCM) – allows a master to control/manage remote processors: power on/off, reset, load firmware
- Inter-Processor Communication (IPC) for shared memory management when sending/receiving data from/to master/remote
- Proxy operations – Remote access to systems services. A transparent interface to remote contexts from Linux user space applications running on the master processor
- libmetal provides an OS environment and hardware abstraction layer
    - Used by the other components of OpenAMP
- Ongoing work to decouple RemoteProc and RPMsg so that they can be used independently

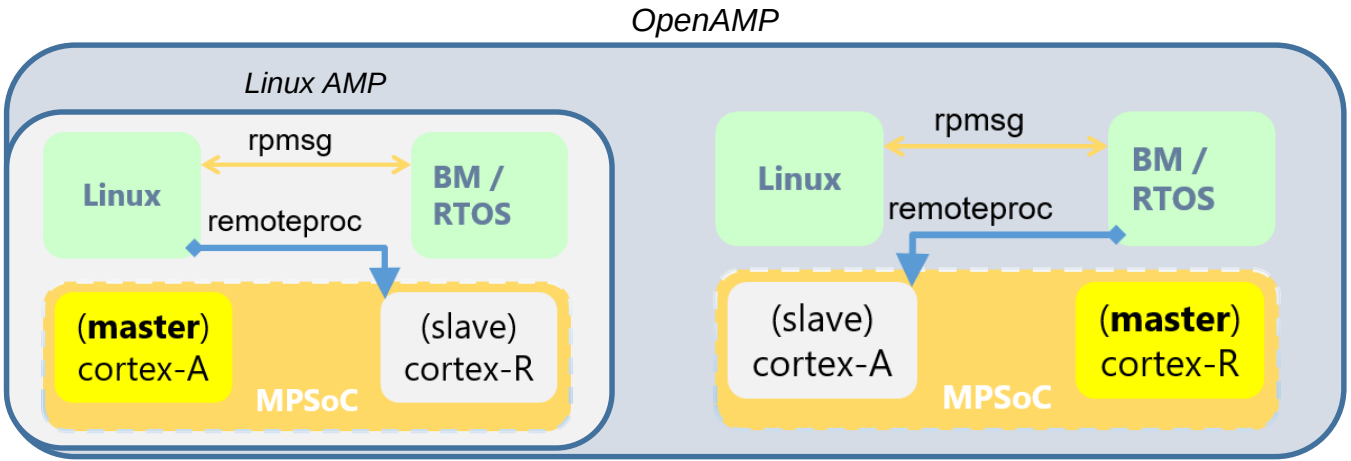
From Linux AMP to OpenAMP

OpenAMP

Linux AMP

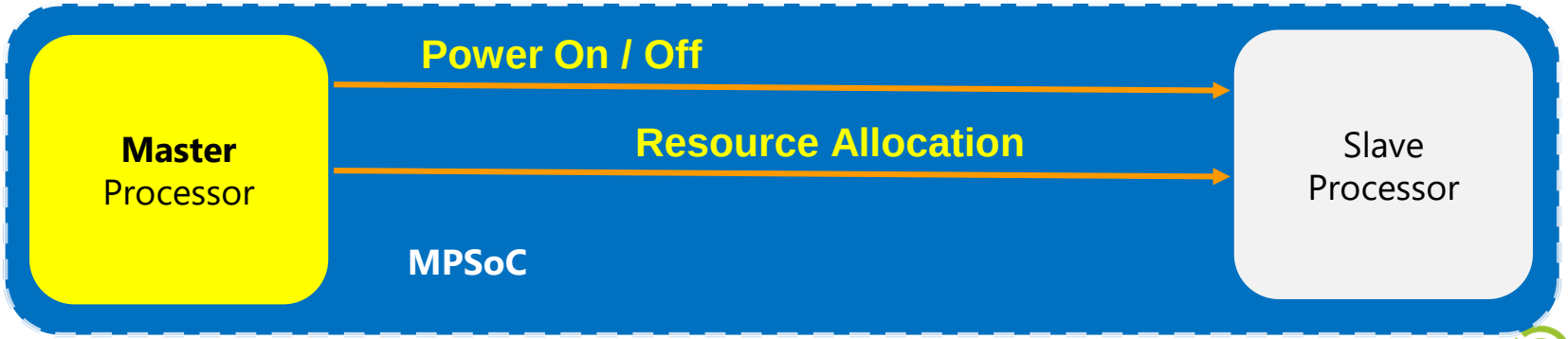
- Kernel modules. No support for firmware on remote processors. Apps on a remote must understand rpmsg/remoteproc.
- Masters must run Linux
- Low level device-specific code is not supported

- User libraries. Adds support for the RemoteProc and RPMsg to RTOS and bare metal
- Master no longer needs to be Linux-based
- Abstracts the device-specific behavior



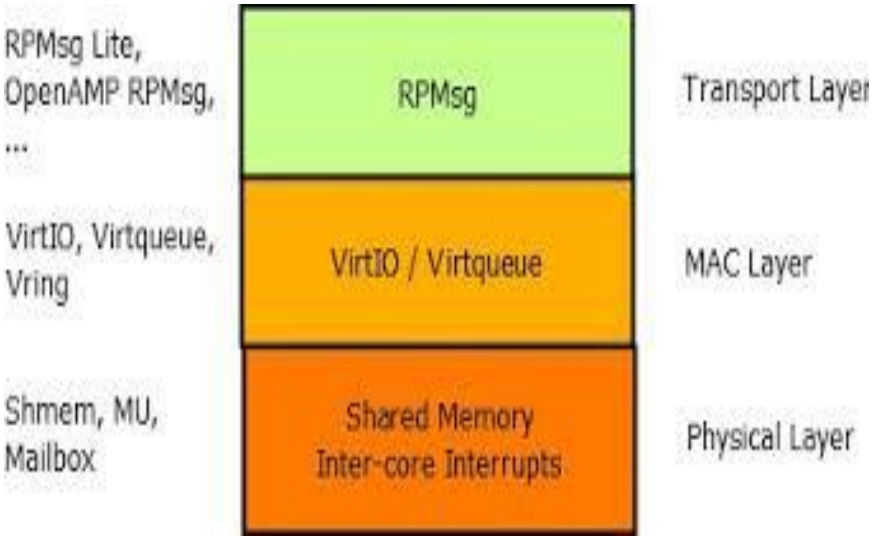
Remote LCM with RemoteProc

- **RemoteProc** – **Remote Processor,** provides support for a master to run firmware on a remote processor.
- **RemoteProc** is a framework that allows a master to control/manage remote processors (power on/off, reset, and load firmware). A RemoteProc driver is used for lifecycle management of remote firmware.
- **RemoteProc**
  - Provides API to control remote processor
  - Abstracts hardware differences between involved processors
  - Establishes communication channels between master and remote processors using the RPMsg framework
  - Declares a minimal set of device-specific low-level handlers



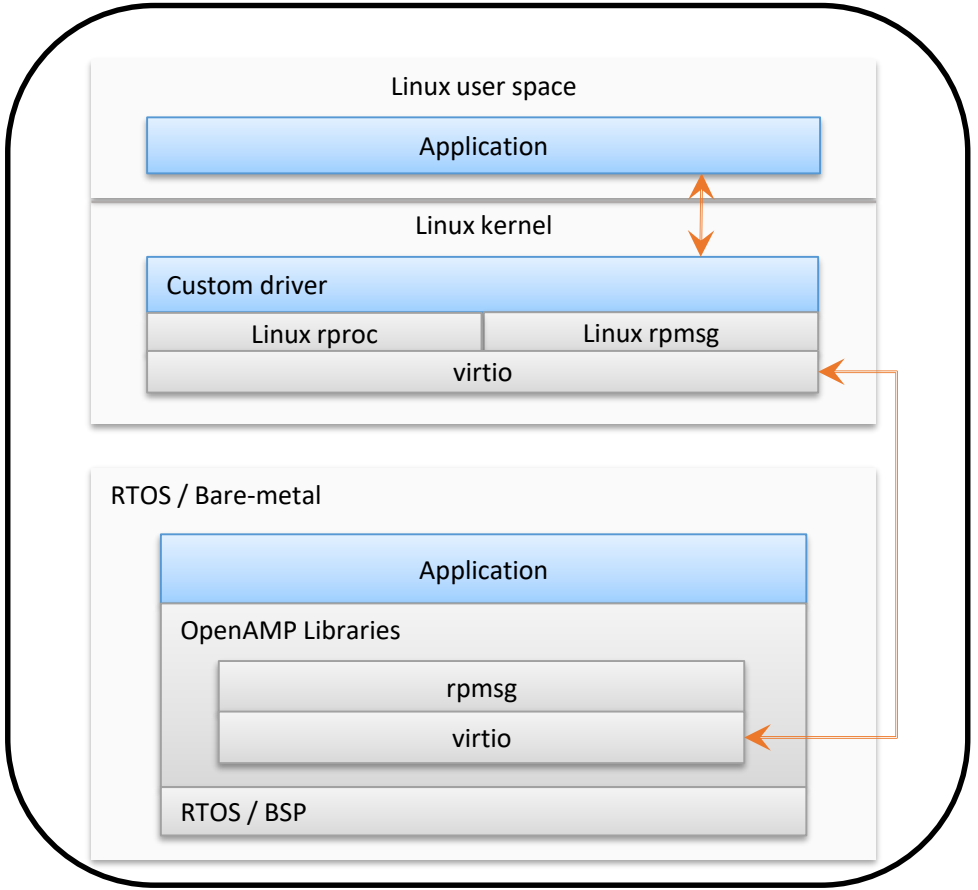
Master/Remote IPC with RPMsg

- RPMsg (Remote Processor Messaging)
  - Provides inter-processor communication (IPC) between master and remote processors.
    - An RPMsg represents a communication channel between the master and a specific remote processor
  - Defines only vendor agnostic aspects of communication
    - E.g. API and the format of messages.
    - Relies on RemoteProc for device-specific handlers



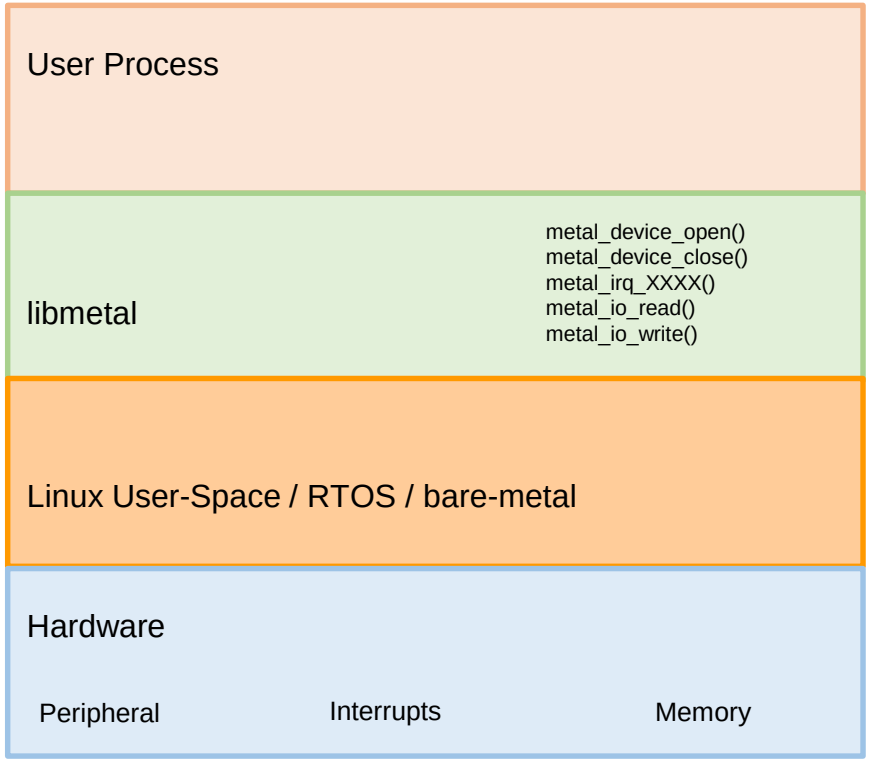
VirtIO

- VirtIO library
  - An abstraction layer over devices in a para-virtualized hypervisor
  - Implements the OASIS virtIO standard for shared memory management
  - A virtualization standard for network, disk device (etc.) drivers
    - Applicable to OpenAMP configurations



Libmetal Overview

- libmetal
  - Provides common APIs for:
    - Device access
    - Interrupt handling
    - Memory management
    - Synchronization primitives
  - Across:
    - Linux user space (based on UIO and VFIO support in the kernel)
    - RTOS (with and without virtual memory)
    - Bare-metal environments
- Fundamental to OpenAMP architecture
  - RemoteProc, RPMsg and VirtIO use libmetal



OpenAMP Remote Startup Process

- OpenAMP architecture
  - Assumes the master is already running and remote processor is in standby or powered down
- Remote Processor Firmware Loading
  - OpenAMP master loads firmware into the memory location
- Remote Processor Start
  - OpenAMP master starts remote processor
    - Example: wake-up remote, release remote from reset, power-on remote, etc.
  - Master waits for remote
  - Master establishes a communication channel to the remote processor

SoC and OS Vendor Support

- Vendor handles the low-level porting for their specific platform(s)
- Vendor supplies example applications for their platform(s)
  - Application includes demonstration of resource table
  - Example application demonstrating basic IPC (e.g., echo)
  - Example application demonstrating master off-loading
- Vendor supplies Linux RPMsg driver for their platform(s)
- Vendor supplies example kernel module and user-space application for interacting with remote device

Status

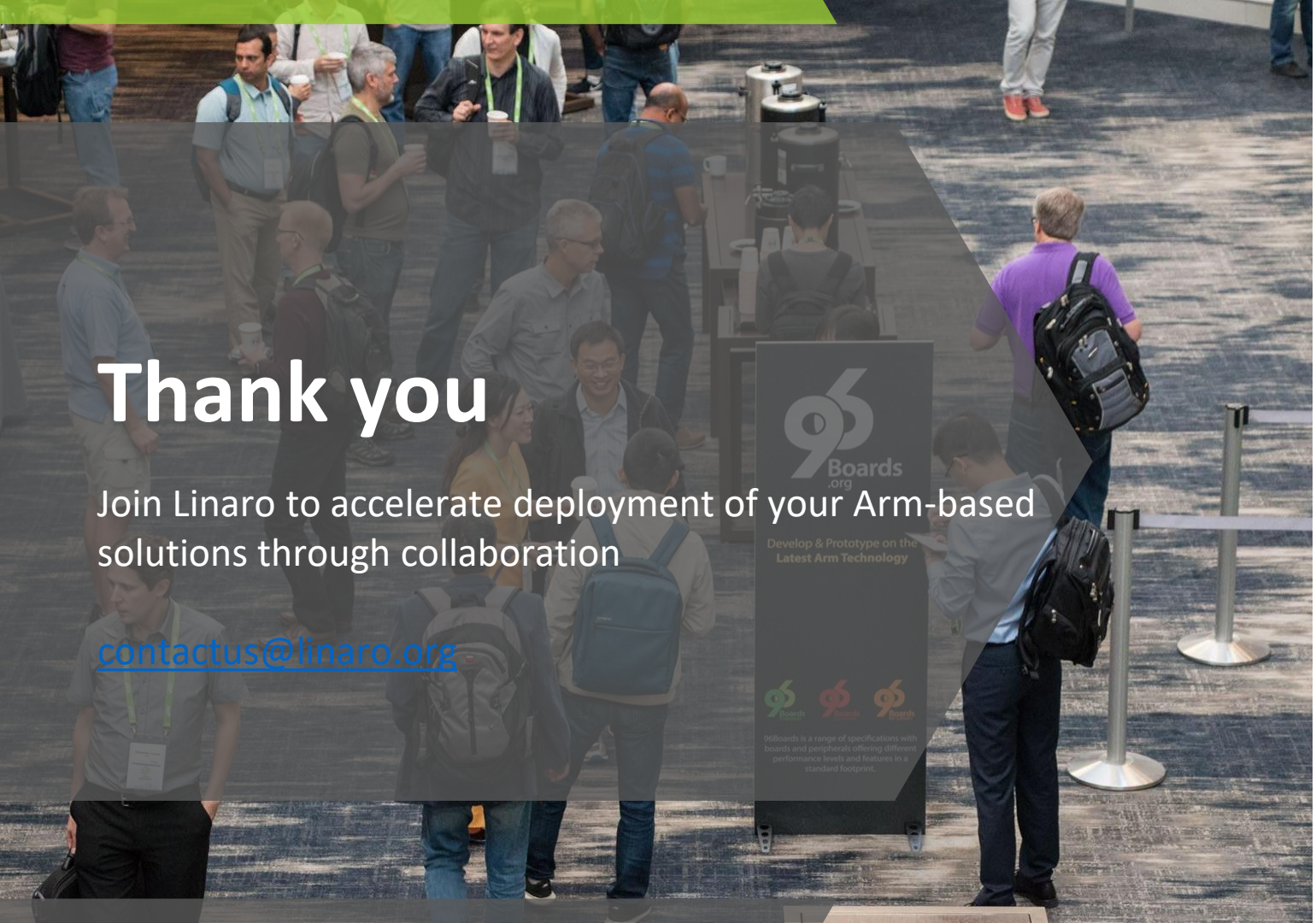
- OpenAMP is an active, evolving community project
  - Project home: github.com/OpenAMP
  - Source structure is fluctuating and standardization is a work in progress
  - Roadmap for advanced use-cases and features
    - IPC performance needs improvement (WIP)
- Availability
  - Commercial
    - uC/OS, Thread-X, Enea OSE, Mentor Nucleus
  - Open Source
    - Zephyr
    - Linux
    - FreeRTOS
  - Porting of the framework still necessary for many commonly used platforms

What's Supported Today

- Range of use cases:
  - Interfaces: message passing, file-system, block, graphics, network
- Provide consistent and portable application interfaces across:
  - Environments: **Linux kernel and user-space**, **FreeRTOS**, **Zephyr**, **bare-metal**
  - Processor architectures: **Cortex-A53,** Cortex-A72, **Cortex-R5, MicroBlaze,** x86, **MIPS32**
  - Secure and **Non-Secure** modes
  - Threads and Processes (on Linux and RTOSes)
  - Virtualized guests and containers (with hypervisors)

Conclusion

- OpenAMP provides a software framework for developers to
  - Enable MPSoC Life Cycle Management (LCM)
  - Load firmware across a multi-processor system
  - Establish communication between the processors
- OpenAMP provides these features in a platform agnostic manner



Thank you

Join Linaro to accelerate deployment of your Arm-based solutions through collaboration

contactus@linaro.org

Linaro connect
Bangkok 2019