NXP | FTF 2016 TECHNOLOGY FORUM

# MULTICORE PROGRAMMING PRACTICES

## FTF-DES-N1838


ROB OSHANA
DIRECTOR SW R&D, MICROCONTROLLER

FTF-DES-N1838
MAY 17, 2016

# AGENDA

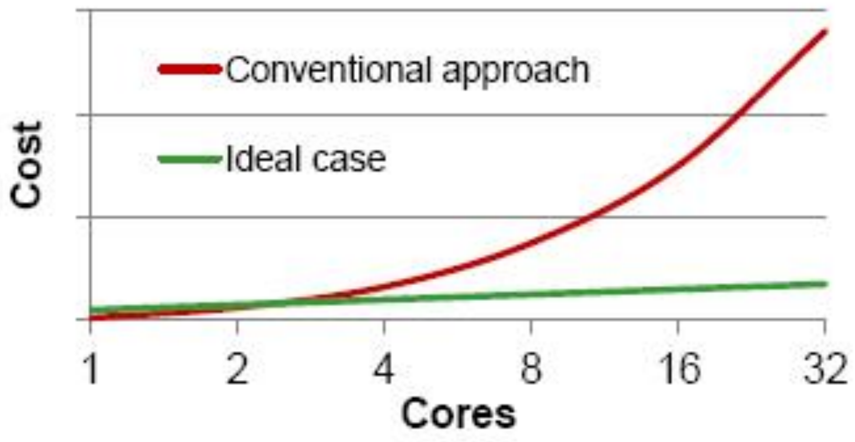
- Multicore Association Update
- Multicore Programming Practices
- openAMP
- Heterogeneous Multicore Benchmarks

# In the Year 2022…

"Multicore has attracted wide attention from the **embedded systems community** […].

However, to obtain good multicore performance, **software is key for decomposing an original sequential program into parallel program parts** and assigning them to processor cores.

So far, such parallelization has been performed by application programmers, but it is **very difficult, takes a long time, and has a high cost**."





H. Alkhatib, P. Faraboschi, E. Frachtenberg, H. Kasahara, D. Lange, P. Laplante, A. Merchant, D. Milojicic, and K. Schwan. *IEEE CS 2022 Report*. IEEE Computer Society, 2014.
www.computer.org/cms/Computer.org/ComputingNow/2022Report.pdf

# Sequential Programming is Kinda Easy…..

## Dot product (sequential)

```
#define SIZE 1000

main() {
  double a[SIZE], b[SIZE];
  // Compute a and b ...
  double sum = 0.0;
  for(int i = 0; i < SIZE; i++)
    sum += a[i] * b[i];
  // Use sum ...
}
```

# Same Algorithm With Multicore Threads… Really??

## Dot product (POSIX threads)

```
#include <iostream>
#include <pthread.h>

#define THREADS 4
#define SIZE 1000

using namespace std;

double a[SIZE], b[SIZE], sum;

pthread_mutex_t mutex_sum;

void *dotprod(void *arg) {
  int my_id = (int)arg;
  int my_first = my_id * SIZE/THREADS;
  int my_last = (my_id + 1) * SIZE/THREADS;

  double partial_sum = 0;
  for(int i = my_first; i < my_last && i < SIZE; i++)
    partial_sum += a[i] * b[i];

  pthread_mutex_lock(&mutex_sum);
  sum += partial_sum;
  pthread_mutex_unlock(&mutex_sum);

  pthread_exit((void*)0);
}
```

```
int main(int argc, char *argv[]) {
  // Compute a and b ...

  pthread_attr_t attr;
  pthread_t threads[THREADS];

  pthread_mutex_init(&mutex_sum, NULL);
  pthread_attr_init(&attr);
  pthread_attr_setdetachstate(&attr,
    PTHREAD_CREATE_JOINABLE);

  sum = 0;
  for(int i = 0; i < THREADS; i++)
    pthread_create(&threads[i], &attr, dotprod,
                   (void*)i);

  pthread_attr_destroy(&attr);

  int status;
  for(int i = 0; i < THREADS; i++)
    pthread_join(threads[i], (void**)&status);

  // Use sum ...

  pthread_mutex_destroy(&mutex_sum);
  pthread_exit(NULL);
}
```

Barbara Chapman, Gabriele Jost, Ruud van der Pas. *Using OpenMP: Portable Shared Memory Parallel Programming*. MIT Press, 2007.

# The Multicore Association

THE Multicore® ASSOCIATION

# The Multicore Association

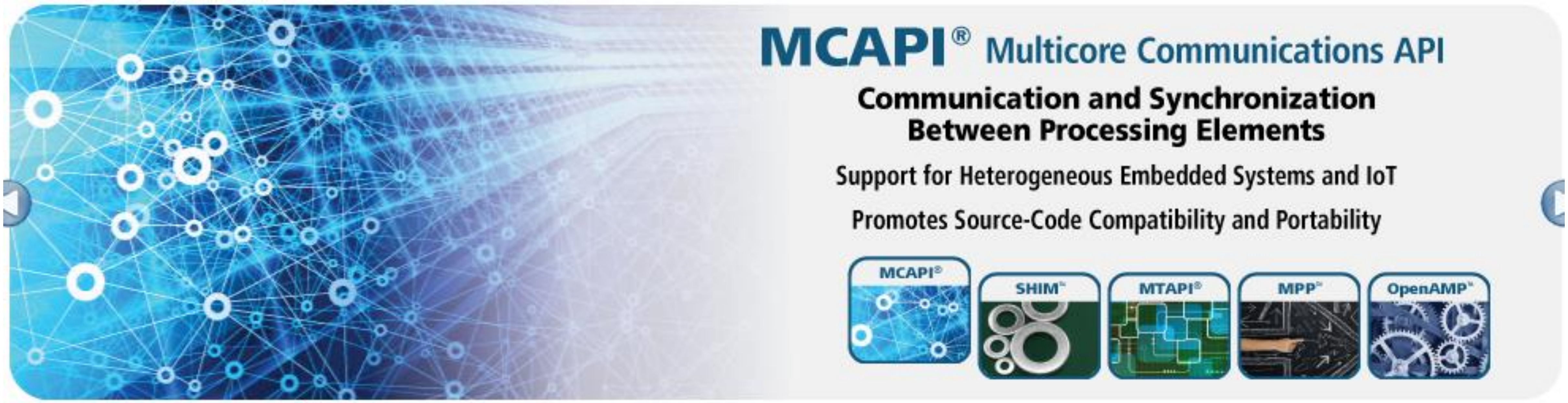
# MPP Status

- 5500+ downloads
- Version 2 working group kicked off Feb 2016

- Data structures and appropriate synchronization strategies (coarse-grained locking, fine-grained locking, lock-free, etc.)
- More material on heterogeneous (for embedded, etc), including mapping concepts for heterogeneous
- Safety critical functionality
- Expand beyond pthreads, add C11/C++11, consider a separate chapter on “Libraries and Foundations”
- Embedded multicore building blocks
- More content about profiling
- Expand “performance” to include power/energy, not just CPU
- Model based development
- Add case studies

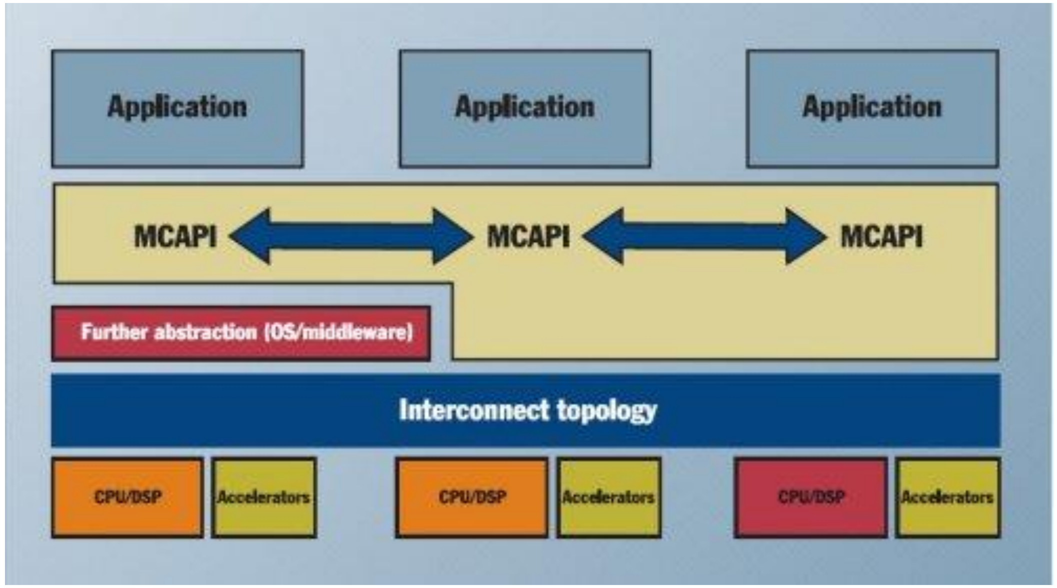
THE Multicore ASSOCIATION™

# MCAPI – Multicore Communications API

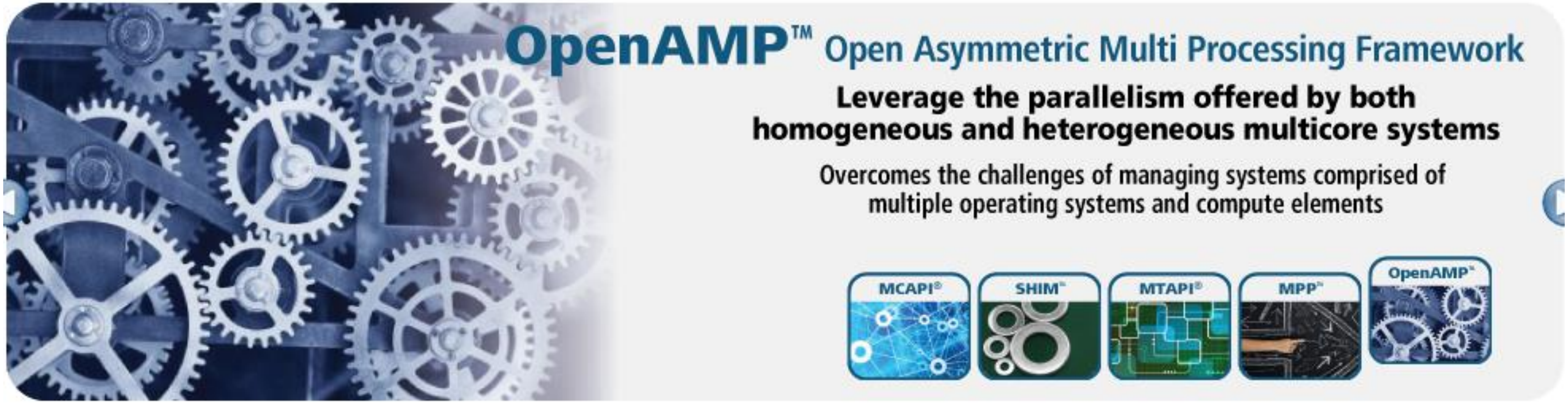


# MCAPI Version 3

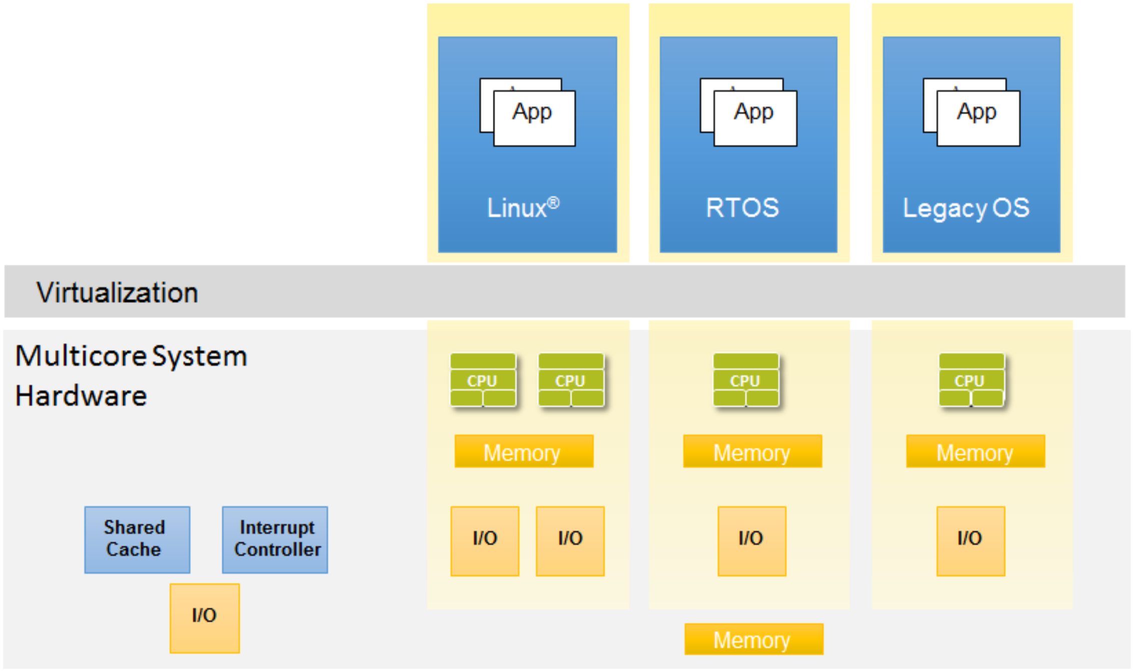
- The MCAPI working group is currently working on version 3.0
- Functional areas that are being worked on include
  - shared memory/zero copy message/packet management functions
  - Interoperability
  - MCAPI subsets, one of which will address the Internet of Things (IoT).
- Chairman; Sven Brehmer, President, PolyCore Software

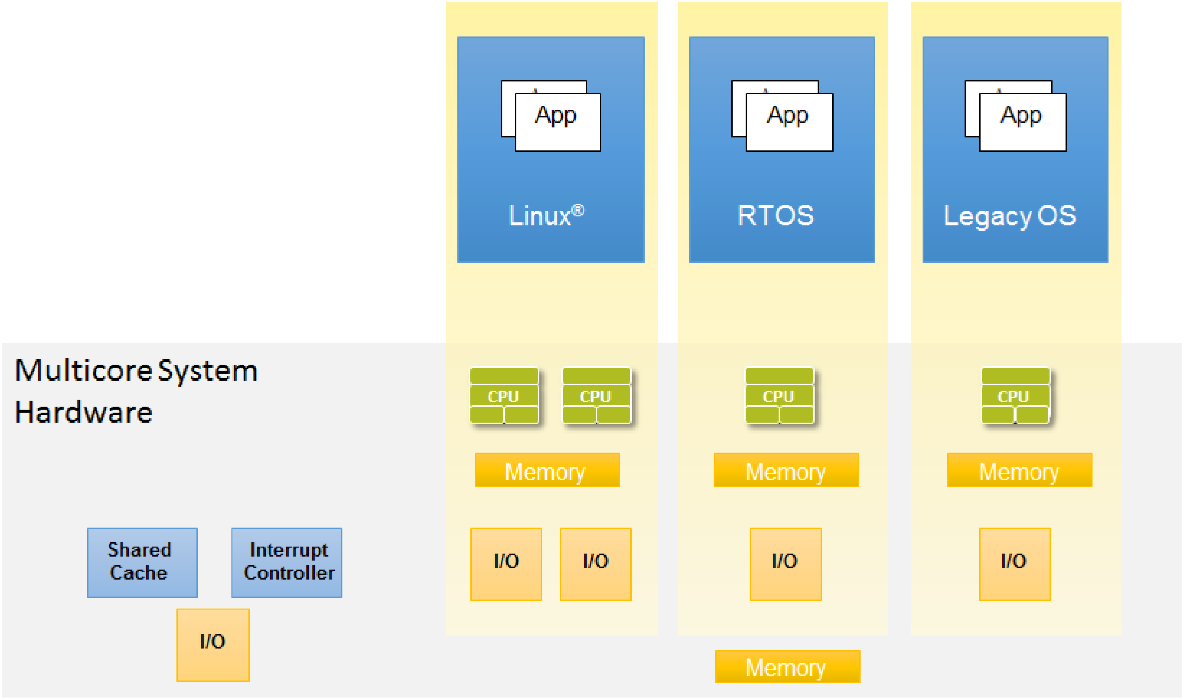


# OpenAMP


OpenAMP™ Open Asymmetric Multi Processing Framework
Leverage the parallelism offered by both homogeneous and heterogeneous multicore systems
Overcomes the challenges of managing systems comprised of multiple operating systems and compute elements
MCAPI®
SHIM™
MTAPI®
MPP™
OpenAMP™

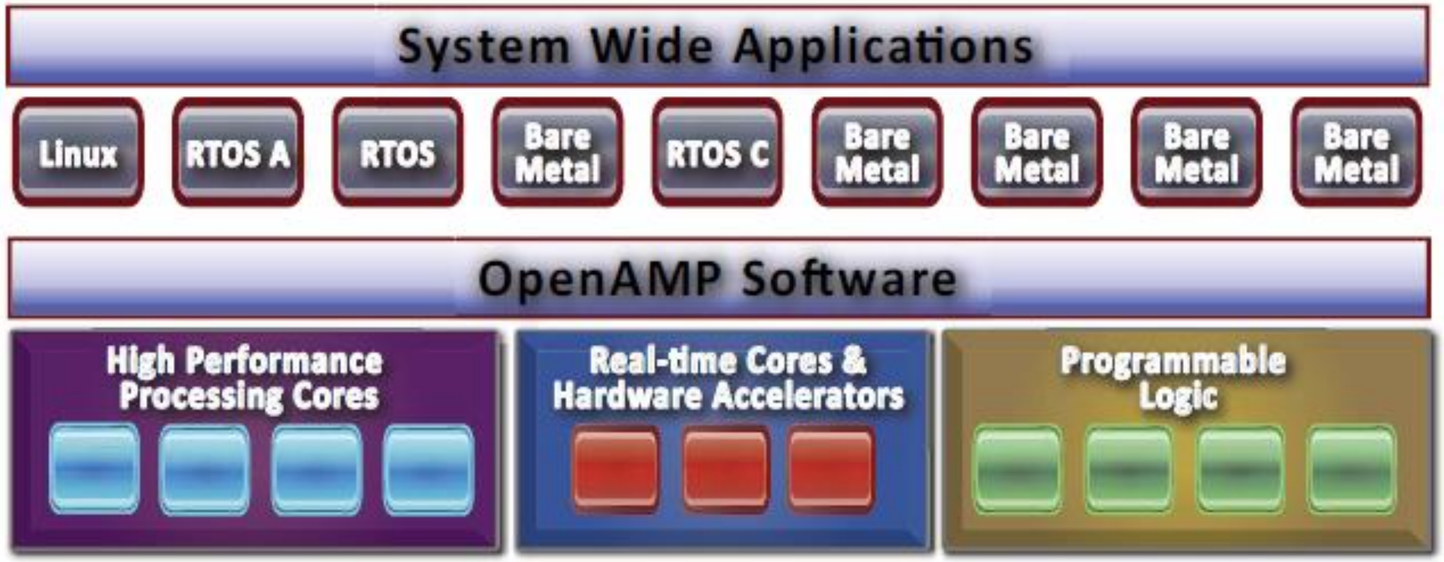
# AMP; Supervised or Unsupervised



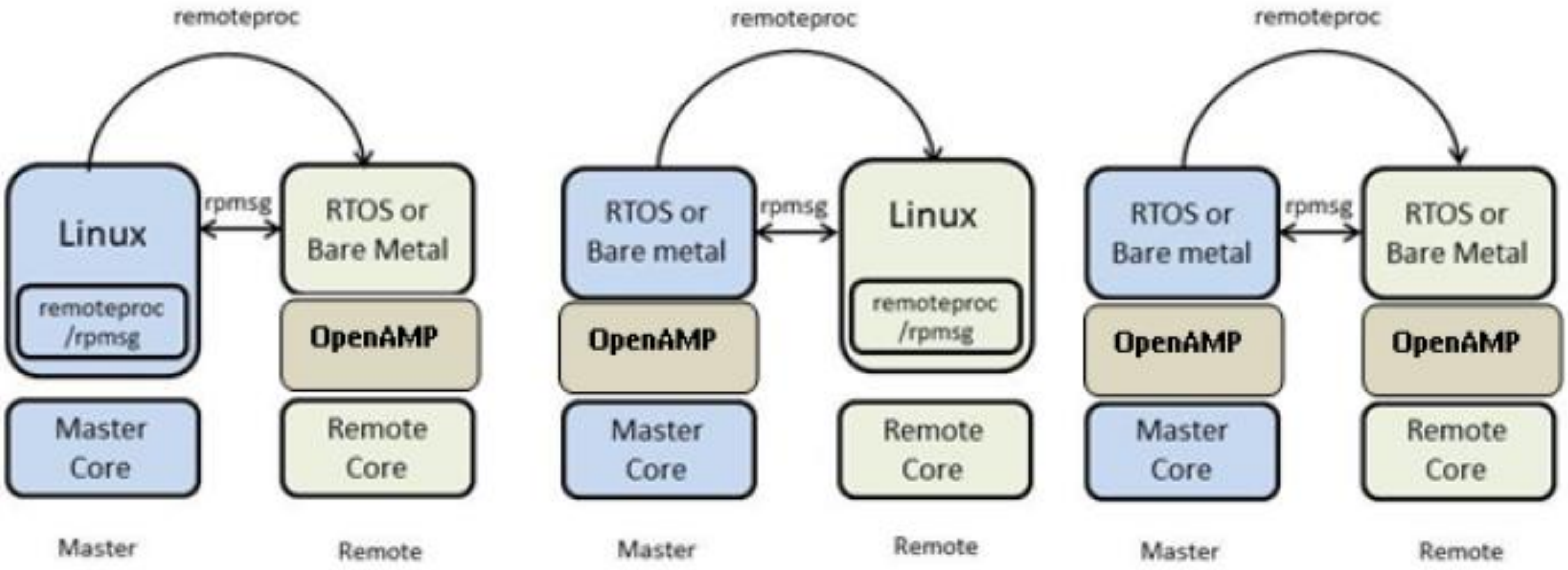



# openAMP

- Life Cycle Management (power on, load firmware, power off, etc) of remote processors from software running on a master processor using remoteproc
- Inter Processor Communication (IPC) between independent software contexts running on homogeneous or heterogeneous cores present in an AMP system using RPMsg
- OSS project within MCA



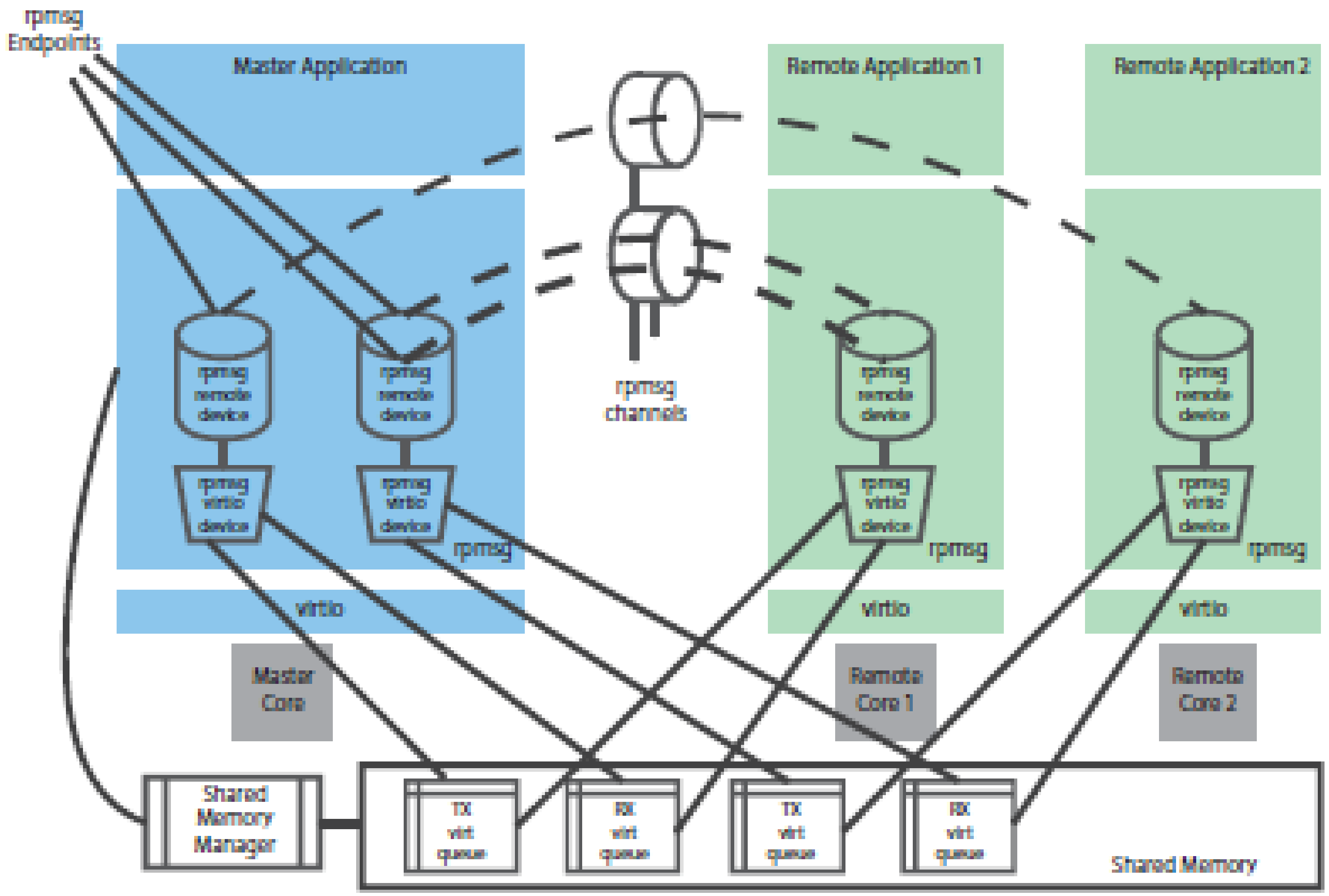
# Typical Topologies/Architectures Found in AMP Systems




**OpenAMP Framework User Reference, Mentor Graphics**

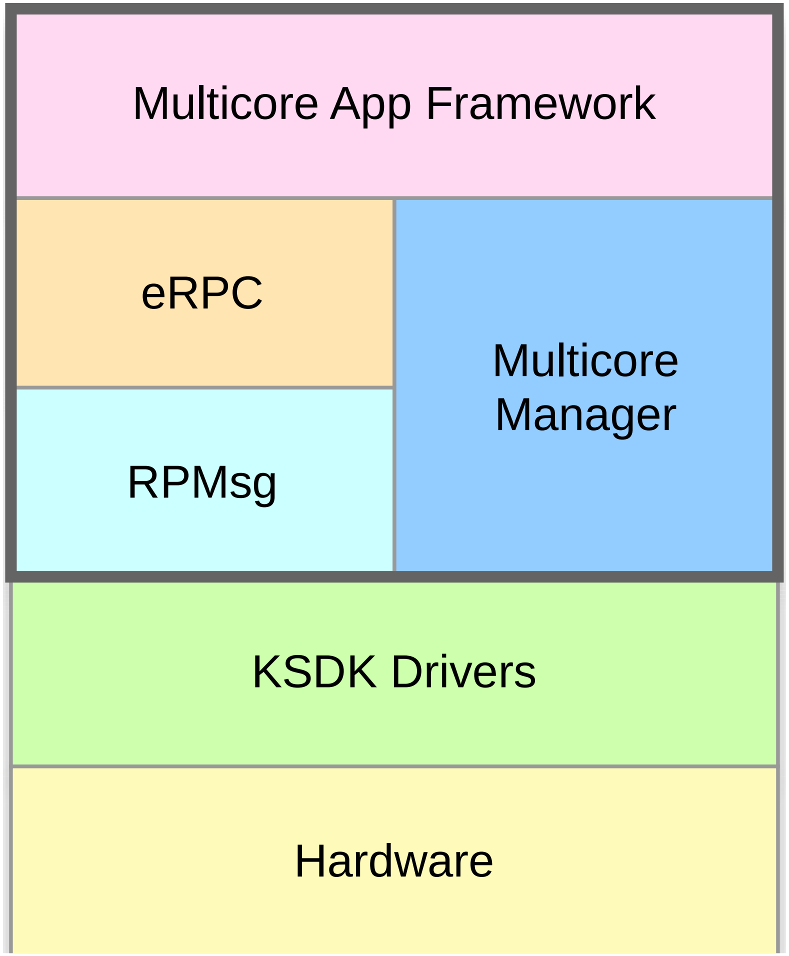
# Use Standard Linux Process Concept to Manage Other OS's

- RPMsg is a virtio-based messaging bus that allows kernel drivers to communicate with remote processors available on the system
- virtIO is an abstraction layer over devices in a paravirtualized hypervisor
- It's the framework used for point to point comms, map physical memory directly into user space applications.
- Every RPMsg device is a communication "channel" with a remote processor

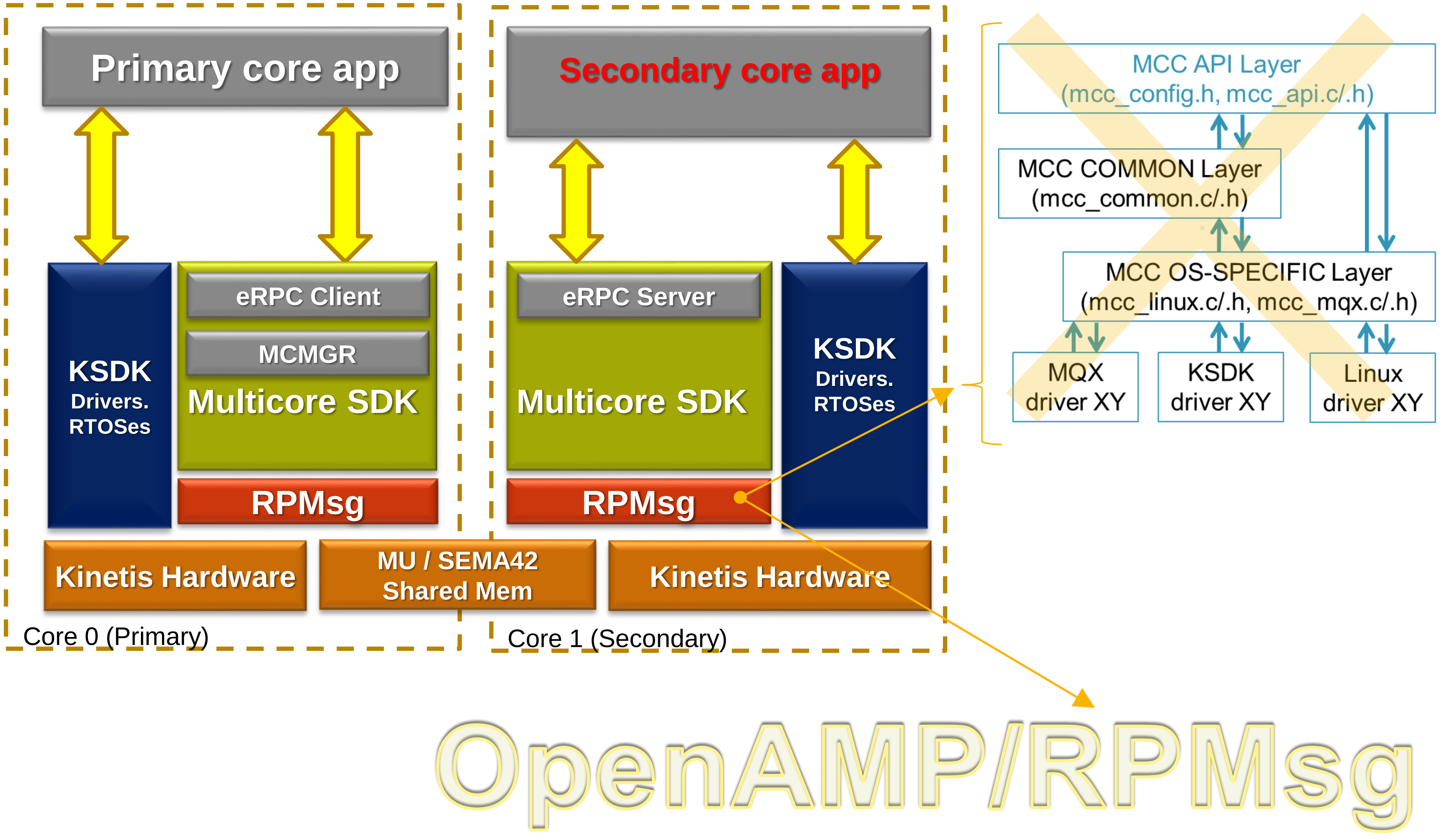


# Multicore Software Development Kit (MCSDK)

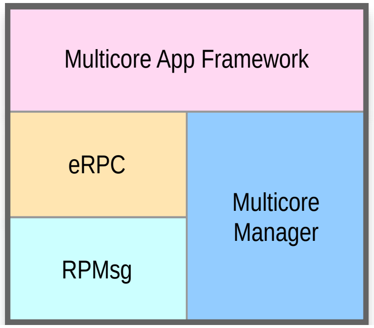
- Multicore Manager (MCMGR)
  - Control and status for cores
  - Domain controller configuration
- Remote Processor Messaging (RPMsg)
  - Inter-core messaging system
- Embedded Remote Procedure Call (eRPC)
  - Provides transparent function call interface to remote services
- Multicore App Framework (MAFMK)
  - Framework for building coprocessor applications



# Multicore Software Framework



# Multicore App Framework




- Framework for building coprocessor apps
- Provides common interface to an app

- Functions:
  - Initialize
  - Sleep
  - Wake
  - Shutdown
  - Clock management
  - Error reporting

```
mcapp_start_app(&appInfo);
mcapp_sleep(policy, allowExternalWake);
mcapp_wake();
mcapp_shutdown();
mcapp_notify_clock_before_change(&clockInfo);
mcapp_notify_clock_after_change(&clockInfo);
```

- Uses eRPC to implement communications
- Provides library on primary and secondary sides for ease of use
- Customers interact with this layer to start/control secondary-core-applications (one function call to start an app.)

# What Type of Applications Could Be Implemented on the Secondary Core and Managed by the MCSDK?

- Communication stacks
  - USB
  - wireless connectivity (Thread, BLE, Zigbee)
- Sensor aggregation/fusion apps.
- Encryption algorithms
- Virtual peripherals
  - UART, SPI, I2C, etc.
- and more…

# OpenAMP Status

- OpenAMP community project & MCA working group established
  - https://github.com/OpenAMP
  - http://www.multicore-association.org/workgroup/oamp.php
  - Initial contributions from Xilinx, Mentor, NXP

- OpenAMP 2016.04 Release Status
  - Currently in code freeze, release planned for end of this month
  - OpenAMP re-structured for open source release
  - Linux libmetal developed and contributed to open source

# OpenAMP 2016 Outlook

- Linux Kernel
  - SMC interface for remoteproc firmware load
    - Allows non-secure world Linux to load and execute a secure world application
      - E.g. Linux non-secure loading a secure R5
      - E.g. Linux non-secure loading a secure EL1-S payload
    - MCA WG could help in specifying lifecycle mgmt. interfaces (impl. via SMC on aarch64)
  - Device tree overlay support for remoteproc
    - One possible approach to decouple remoteproc from rpmsg in the kernel
    - High priority use-case for Xilinx – allow use of kernel remoteproc paired with user-space rpmsg
    - Approach is to use device-tree overlays to register remote devices instead of resource table discovery
  - IOMMU support for ZynqMP remoteproc
    - Allow system (i.e. Linux managed) memory to be used for remote processor execution with IOMMU translation

# OpenAMP 2016 Outlook

- LibMetal
  - VFIO support
    - Extend Linux I/O framework to include VFIO and IOMMU support
  - DMA map/unmap operation
    - Could use MCA WG review of DMA-API abstractions in LibMetal for various system models (i.e. coherent/non-coherent, with/without IOMMU, etc.)
  - Timer operations
    - Addition of time-keeping interfaces into libmetal
    - Could use MCA WG help refining API definitions
  - FreeRTOS support

# OpenAMP 2016 Outlook

- OpenAMP
  - OpenAMP port to libmetal
    - Enable Linux user-space rpmsg operation
    - MCA WG could help review libmetal interfaces to ensure general applicability across Linux/RTOS/BM/...
  - Zero copy support
    - Need feature negotiation mechanisms to prevent backward compatibility issues
    - MCA WG could help review rpmsg interface extensions for zero-copy operation
  - Polled mode in rpmsg
  - 64-bit address space support
  - Inter-process rpmsg; Built-in self tests
  - Application owned object allocation

# Embedded Microprocessor Benchmark Consortium

- Industry-Standard Benchmarks for Embedded Systems

- EEMBC, an industry alliance, develops benchmarks to help system designers select the optimal processors and understand the performance and energy characteristics of their systems. EEMBC has benchmark suites targeting cloud and big data, mobile devices (for phones and tablets), networking, ultra-low power microcontrollers, the Internet of Things (IoT), digital media, automotive, and other application areas. EEMBC also has benchmarks for general-purpose performance analysis including CoreMark, MultiBench (multicore), and FPMark (floating-point).

# Industry-Standard Benchmarks for the Embedded Industry

- EEMBC – industry alliance formed in 1997
  - Founded by Markus Levy
- Defining and developing application-specific benchmarks
- Targeting processors and systems
- Expansive Industry Support
  - >45 members
  - >120 commercial licensees
  - >140 university licensees

# Subcommittees and Working Groups

| Subcommittees | Active Working Groups Within Subcommittee |
|---|---|
| Low Power | IoT-Connect<br>IoT Security<br>ULPBench-Peripheral |
| Imaging | Heterogeneous Compute |
| Cloud and Big Data | ScaleMark<br>IoT Gateway |
| Automotive | AutoBench 2.0 |
| Mobile | BrowsingBench 2.0 |
| Processor Performance | AutoBench 2.0 (MITH)<br>CoreMark-FP |

# EEMBC IoT Working Groups

- **IoT-Connect Benchmark**
  - Performance and energy measurement of edge-node devices and microcontrollers and RF components
  - Beta in Q3/16
- **IoT-Security Benchmark and Guidelines**
  - Measuring overhead of implementing security on edge-node (microcontrollers) and gateway devices (SoCs)
  - Performance, energy, memory
  - Alpha Q4/16
- **IoT Gateway Benchmark**
  - In definition stage determining gateway profiles and key metrics

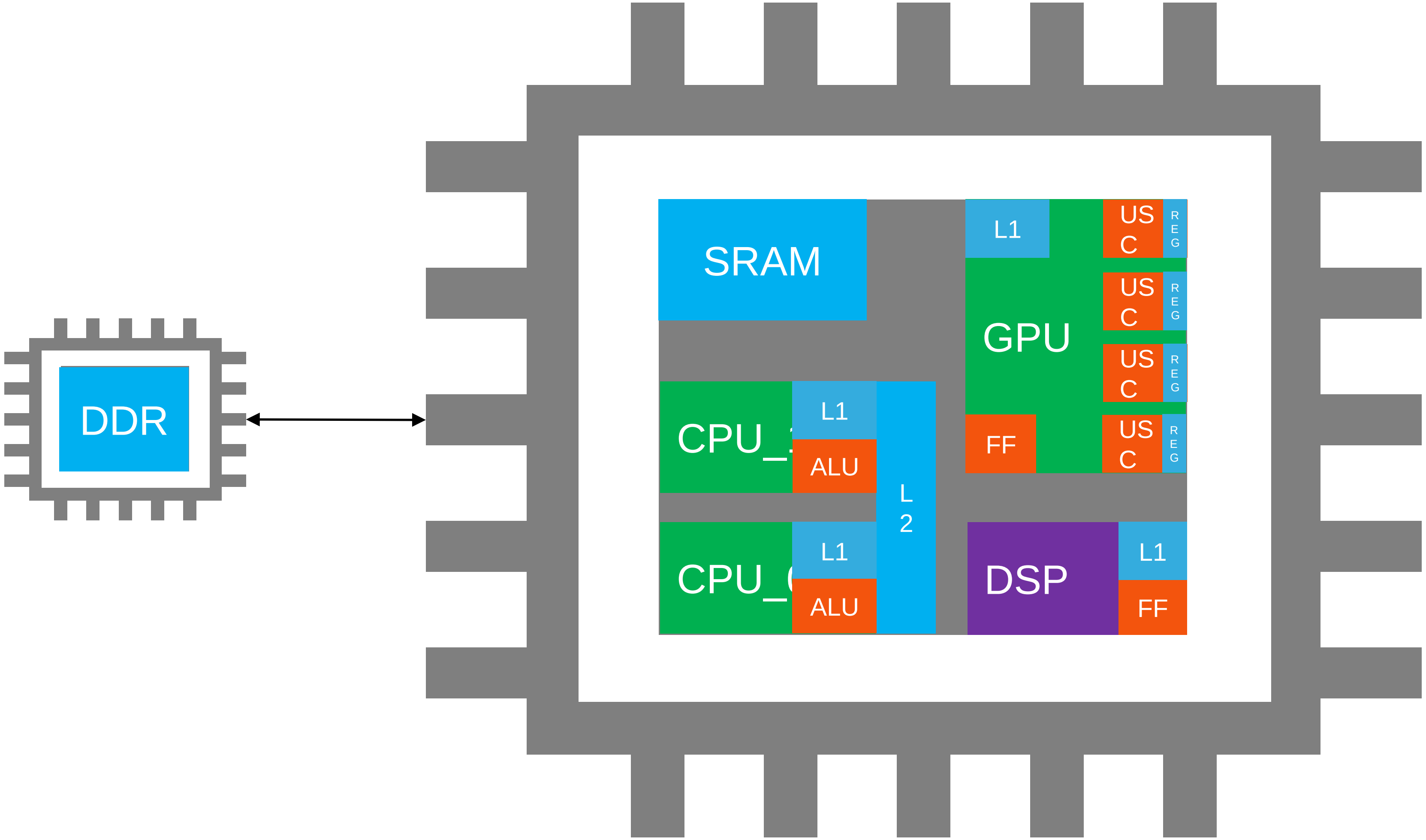
# Heterogeneous Compute Benchmark

- Compute applications benchmark, which also serves as a detailed compute performance analysis tool.
- Targets mobile imaging, autonomous driving, computer vision
- Combines synthetic and applications workloads utilizing OpenCL 1.2 and automatic optimization
- Beginning development phase in Q2/16
- http://www.eembc.org/compute/about.php
- Chairman: Rafal Malewski, NXP Semiconductors

# Heterogeneous Compute

- CPU, GPU, DSP - choice depends on algorithm and data size, varies throughout processing flow
- Synchronization and data transfer latencies complicate the pipeline design.
- Unified Memory Architectures, as on the i.MX applications processors, reduce the need for data transfers between compute devices. DMA from all compute devices to the same physical memory.
- Understanding the architecture of both the individual compute device, and the whole system is key to success.

# Understanding the Puzzle



# Embedded Compute: It Is a Jungle of API’s

## Market trends

| | OpenCV | OpenCL | OpenVX | CUDA | OpenGL Compute | Vulkan Compute | NEON | Proprietary |
|---|---|---|---|---|---|---|---|---|
| CPU | x | x | x | | | x | x | |
| GPU | (x) | x | (x) | x | x | x | | |
| DSP | (x) | (x) | (x) | | | (x) | | x |

- OpenCL is currently the most widely adopted compute API on embedded systems.
- Vulkan Compute with SPIR-V has the potential to become the next gen de-facto standard, but adoption will take years

# Heterogeneous Compute - Benchmarking

Goal is a compute applications benchmark, which also serves as a detailed compute performance analysis tool.

# Existing Compute Benchmarks

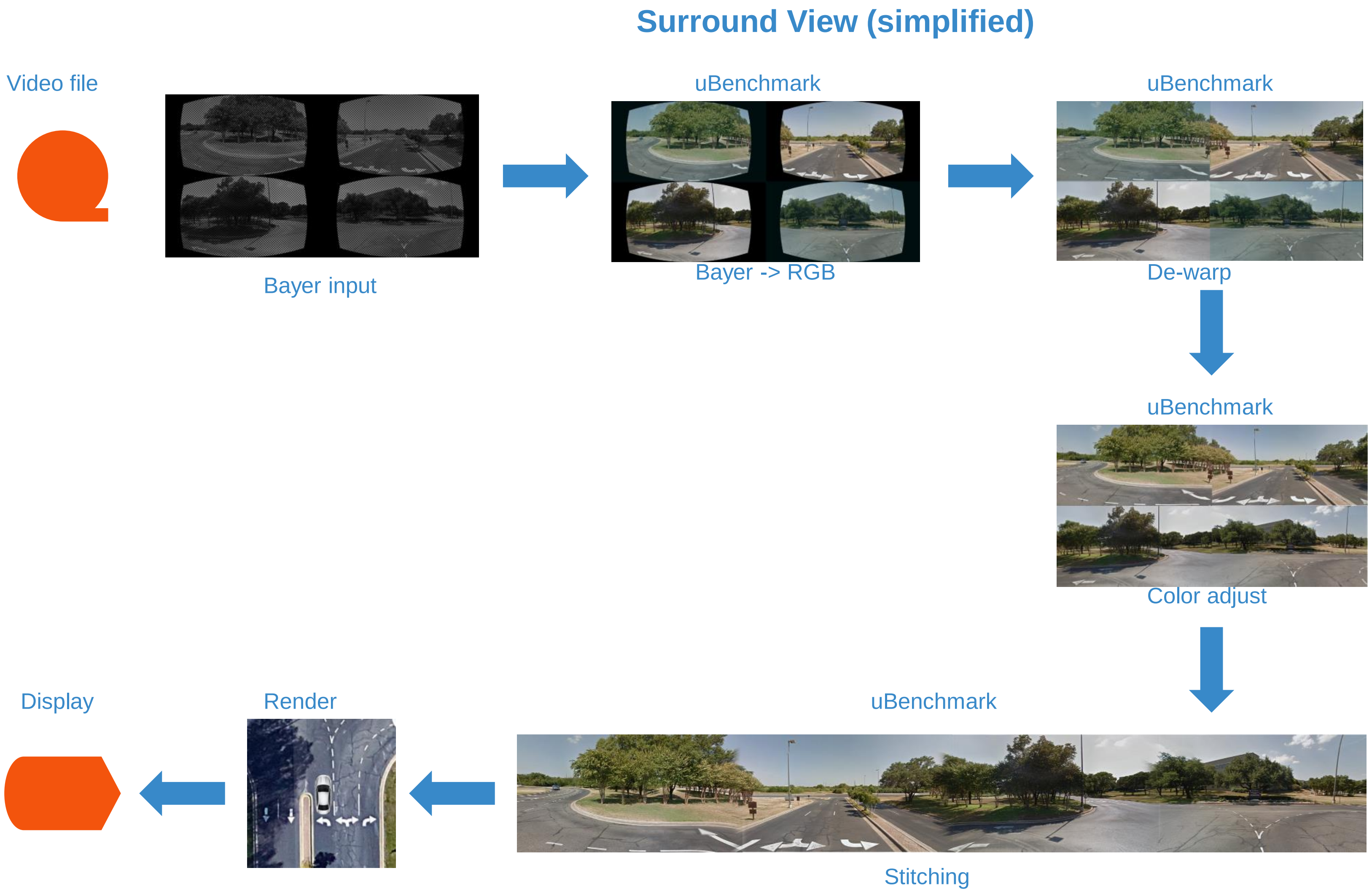
- Existing compute benchmarks fall within two distinct categories
  - Synthetic test collections
    - Isolated tests of specific instruction sets / data paths
    - Results are not indicative of total system performance in compute applications
  - Application use cases
    - Highlevel tests for raytracing, image processing, finances
    - Results can often be misleading, as the application processing flow is deployed in a common way across test platforms, not considering optimal paths

- There is a need for a benchmark which can combine the best of both methodologies

# EEMBC – Compute Benchmark Concept

- A set of applications that represent real products
- Each application use case is broken down into the individual steps which comprise the processing pipe = uBenchmarks, in OpenCL 1.2
- Each uBenchmark shall iteratively be deployed on all possible compute devices on the test system.
- Each uBenchmark shall be optimizable to the compute device it is running on.
- Based on the performance of each uBenchmark, and the combined parallel flow, an optimal processing path can be determined.

# Benchmark Flow Example



# EEMBC – Heterogeneous Compute Benchmark

- NXP is key member of new Working Group
- Automotive Surround View and Mobile Augmented Reality as first application flows
- Initial release by Q4’16

# Summary

- MCA advancing several important multicore concepts/technologies
  - MPP with industry case studies
  - openAMP
  - MCAPI updates for performance and lightweight devices

- EEMBC also participating in multicore
  - Heterogeneous computer benchmarks

# Shameless Plug

- Christmas
- Spouse birthday
- Etc.

# QUESTIONS?

SECURE CONNECTIONS
FOR A SMARTER WORLD

# ATTRIBUTION STATEMENT