Virtual Open Systems

VOSYSmonitor

A Low Latency Monitor Layer for Mixed-Criticality Systems on ARMv8-A

contact@virtualopensystems.com

29th Euromicro Conference on Real-Time Systems
2017-06-28, Dubrovnik, Croatia.

Autorship and sponsorship

Virtual Open Systems is a high-tech software company active in open source virtualization solutions and custom services for complex mixed-criticality automotive, NFV networking infrastructures, consumer electronics, mobile devices and in general for embedded heterogeneous multicore systems around new generation processor architectures.

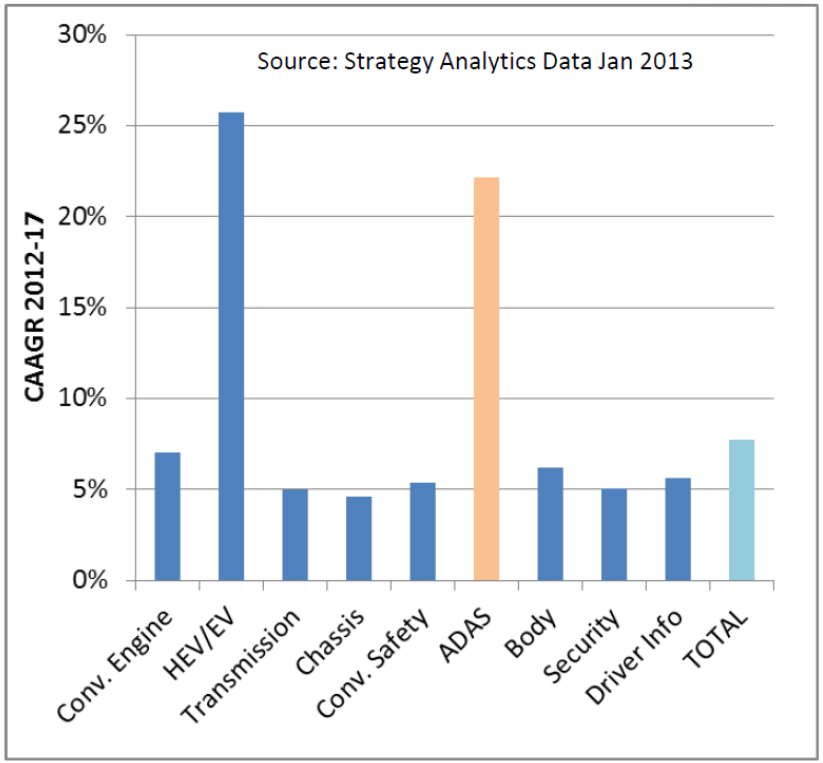
This work is done in the context of the FP7 Distributed REal-time Architecture for Mixed criticality Systems (DREAMS) project under the grant agreement number 610640 (http://www.dreams-project.eu/).

Presentation Plan

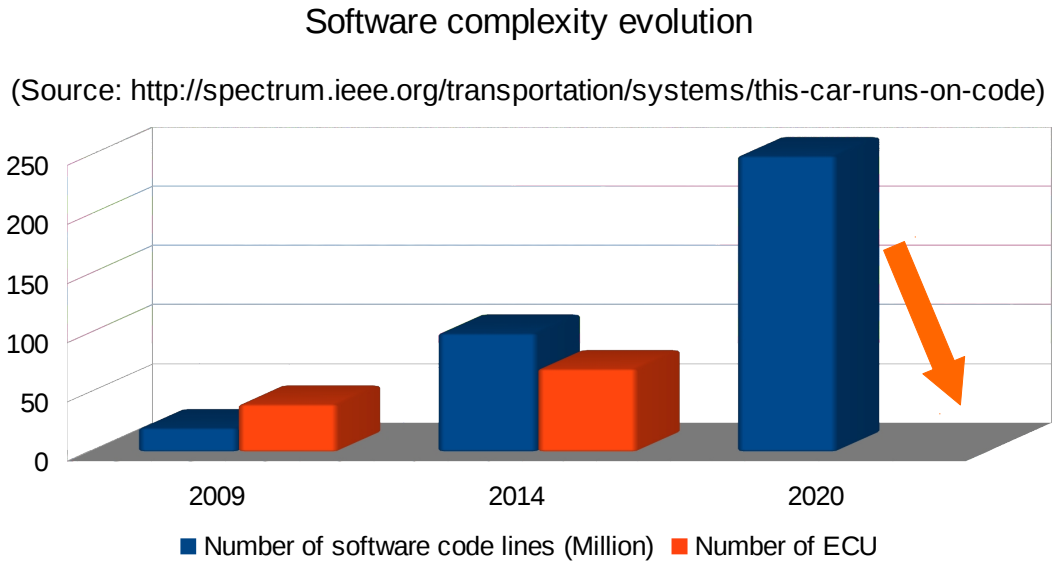
- **Centralized ECUs integration on ARMv8-A**
- VOSYSmonitor for automotive mixed-criticality systems
- Benchmark and conclusion

Growing Areas of Automotive Electronics

Growth Areas – System Types (by Strategic Analytics)



Rapidly growing number of software functions



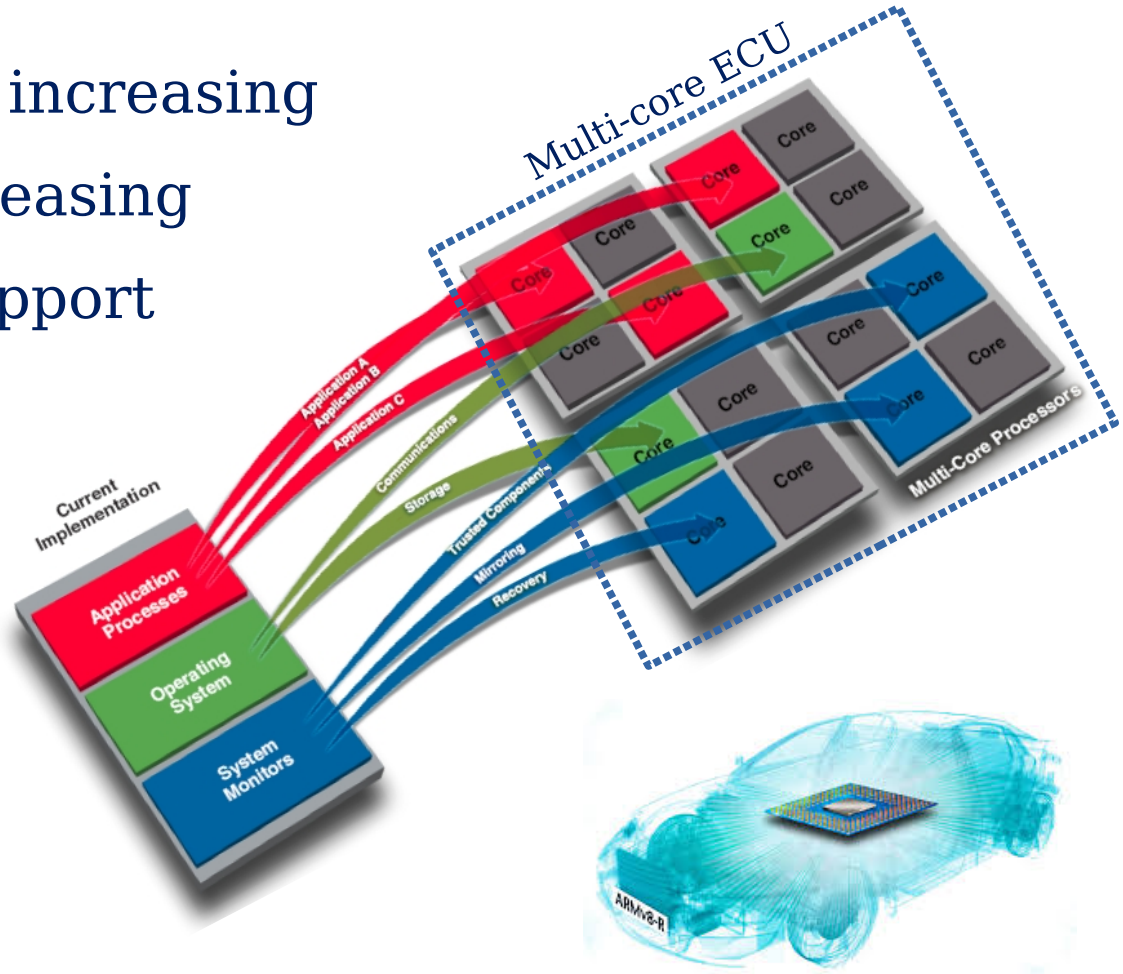
All automotive system areas are growing up, specially systems related to ADAS and HEV/EV. This trend will considerably increase the number of software functions while ECU numbers should be decrease.

Low-power and High-performance Computing for centralized ECUs

Last multi-core architectures are bringing new functionalities to the automotive platforms :

- Computing performance is increasing
- Power consumption is decreasing
- Hardware virtualization support

The main interest is to use the computing performance and hardware capabilities to embed more functionalities, having different levels of criticality, in the same ECU in order to decrease the number of hardware platforms needed in the car.



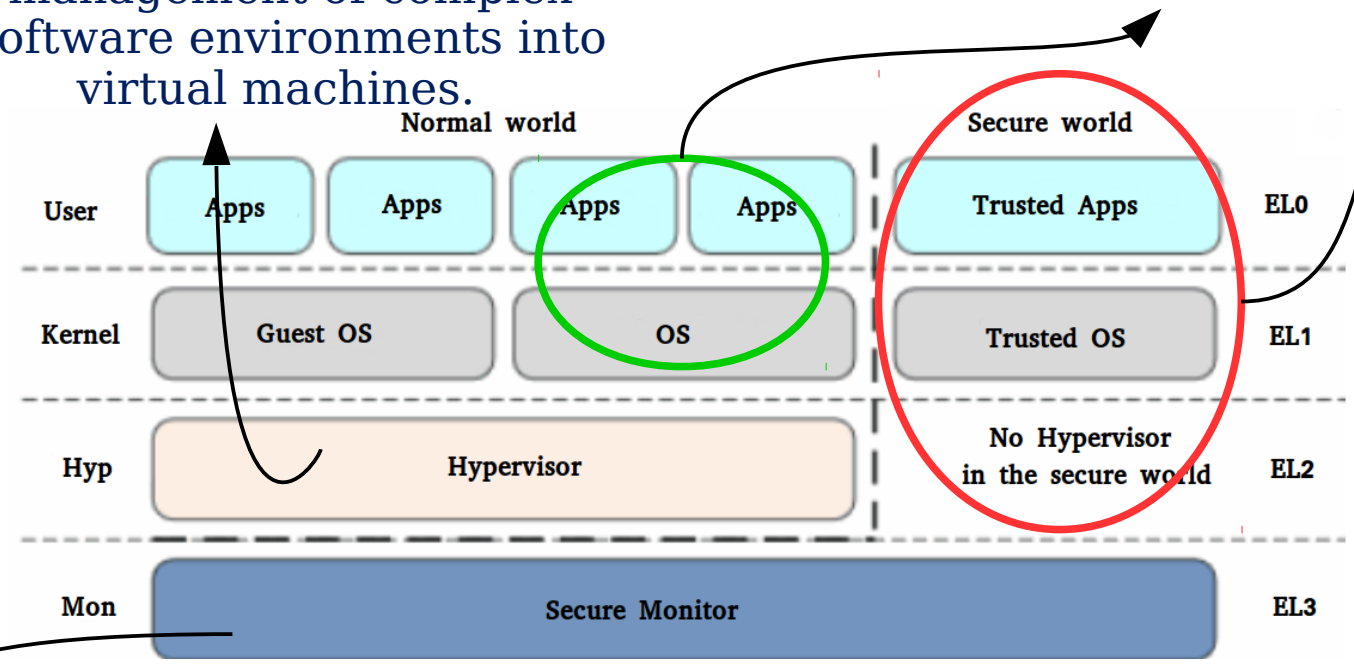
Centralized ECUs integration: challenges

Such a concept of the future car, brings new and unprecedented challenges to the automotive industry:

- **Multi-OS support and integration**: As functions are served by different operating systems (e.g., AUTOSAR for safety-critical functions, GenIVI Linux for automotive infotainment, Android for user apps), the multi-core system needs to be able to run multiple operating systems at the same time.
- **Efficient shared use of SoC resources**: Different functions make use of the same dedicated system resources. Examples for this include accelerated graphics from different integrated functions, or the shared use of communication channels.
- **Separation of functions and mixed-criticality support**: Safety critical functions need to be able to run alongside non-safety-critical functions without compromising their safety characteristics.

ARMv8-A exception level

- ***ARM Virtualization extensions*** address the needs of devices for the partitioning and management of complex software environments into virtual machines.
- ***Normal world*** to run concurrently another OS (e.g Linux) without impacting the secure OS.



- ***Secure world*** is completely isolated (memory, devices, etc) from the Normal world by *ARM TrustZone* security extensions. Since TrustZone is implemented in hardware, it reduces the security vulnerabilities. The secure world could be used to run a secure OS to provide secure services to the OS running in the Normal world.
- ***Monitor layer*** is the highest priority level which provides a bridge between each world to allow some interactions.
- **Exception level changing through specific instructions SMC, SVC, HVC, ERET**

ARMv8-A TrustZone

Hardware security extension

- CPU partitioned in two compartments: Secure and Non-secure worlds.
- Secure world dedicated to Trusted OS and applications.
- Normal world available for rich GPOS with virtualization features (e.g. Liunx KVM).
- MMU TLBs and caches are assigned to each world.
- Interrupts can be divided in secure / non-secure.

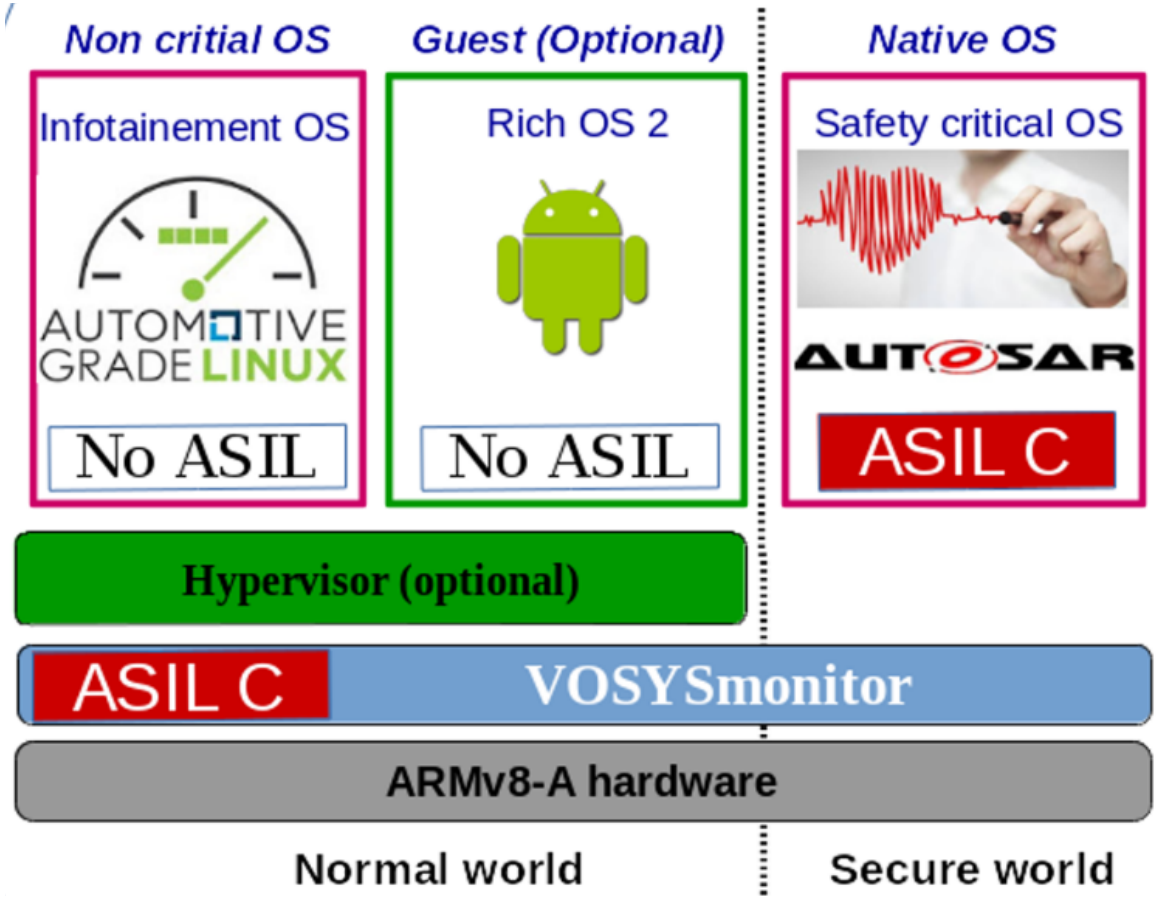
Presentation Plan

- Centralized ECUs integration on ARMv8-A
- **VOSYSmonitor for automotive mixed-criticality systems**
- Conclusion

VOSYSmonitor System Overview

- VOSYSmonitor is a software layer based on ARM Trusted Firmware, which enables the co-execution of virtualized systems along with safety critical applications on the same platform and/or core.

- Safety critical OS isolation using ARM Trustzone
- GPOS virtualization extensions (KVM) enabled
- High priority to the critical applications to meet timing constraints
- When no pending task in the Safety critical OS, the control is given to the non-critical OS

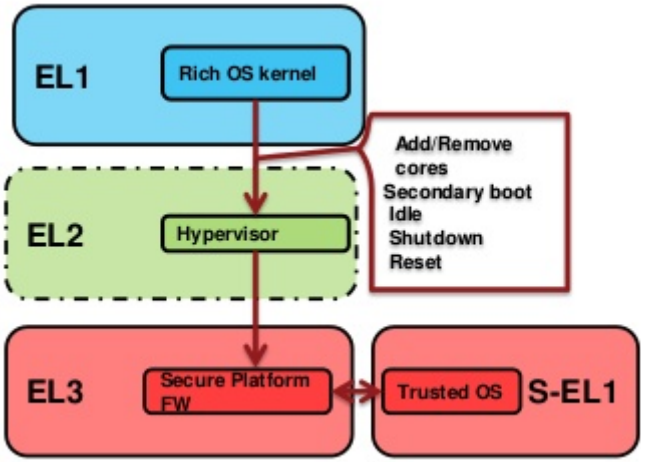


VOSYSmonitor compliance

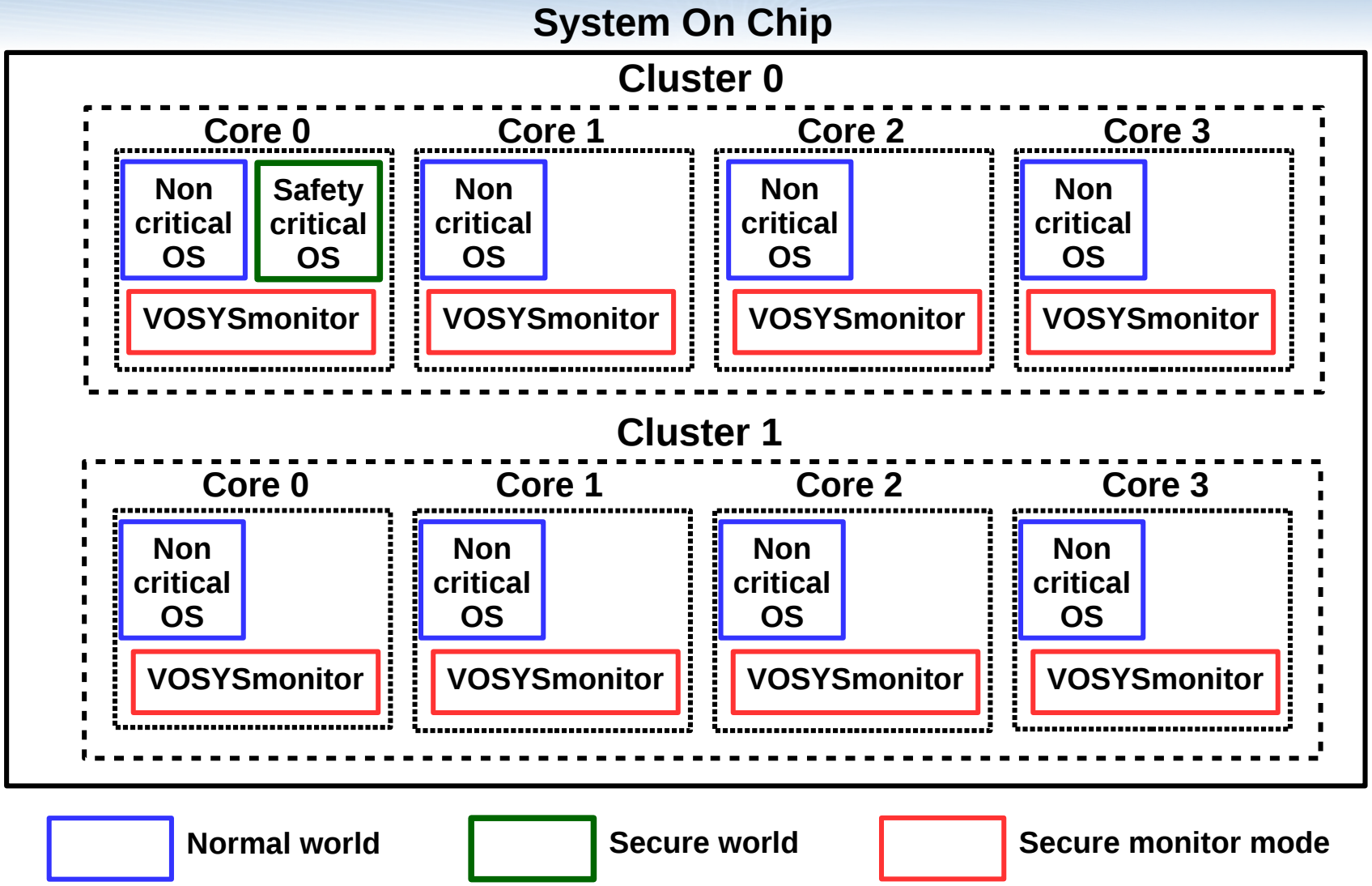
VOSYSmonitor is fully compliant with ARM conventions/protocols :

- SMC Calling convention (SMCCC)
  - ✔ Define a convention for SMC in ARM v7/v8 (e.g., register use for parameters and returns values, SMC type, etc)
  - ✔ Specify a partitioning of service providers to allow the vendors coexistence in the secure firmware (e.g., ARM, OEM, SIP, Trusted OS)

- Power State Coordination Interface (PSCI)
  - ✔ Define a standard interface to handle power management requests.
  - ✔ Define a protocol to allow secure firmware to arbitrate power management requests
  - ✔ Power control method in Linux AArch64 kernel



System overview: multi-core SOC



VOSYSmonitor environment

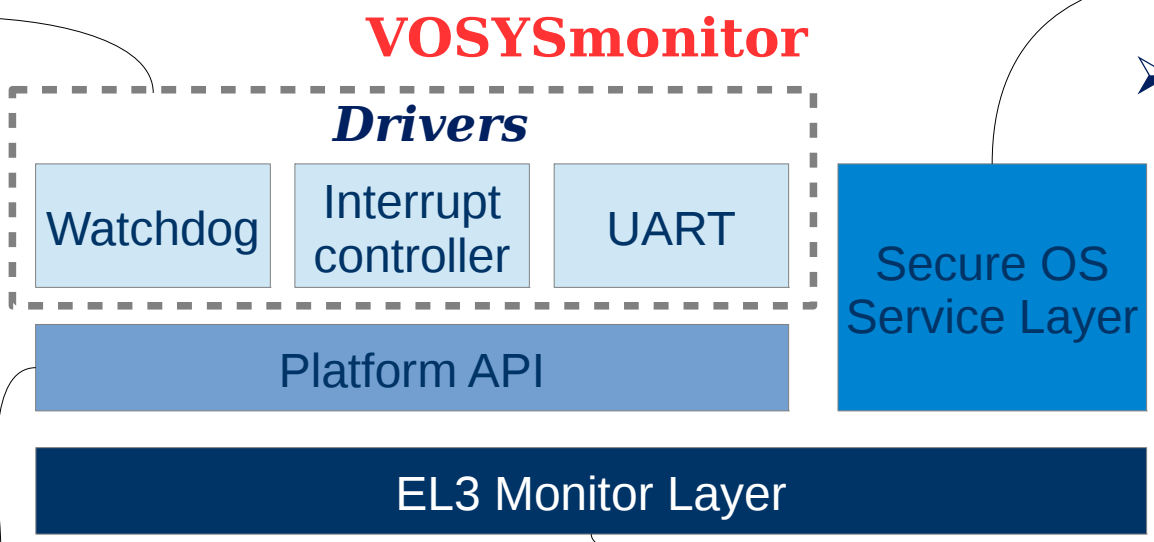
- VOSYSmonitor is compiled with ARM Compiler 6
  - ✔ ARM compiler 6 will be ISO-26262 compliant for the end of Q2 2017 to enable users to apply this compiler for safety-related development without qualification activities.

- VOSYSmonitor has been ported on several ARMv8 development platforms:
  - ✔ ARM Fast Models AEMv8A (Virtual Platform)
  - ✔ ARM JUNO Development board
  - ✔ Renesas R-CAR H3/M3 board Compliant with ISO-26262 (ASIL-B)
  - ✔ Nvidia Jetson Tegra X1
  - ✔ VOSYSmonitor is portable on all ARMv8-A and ARMv7-A boards with TrustZone extensions.

VOSYSmonitor software architecture

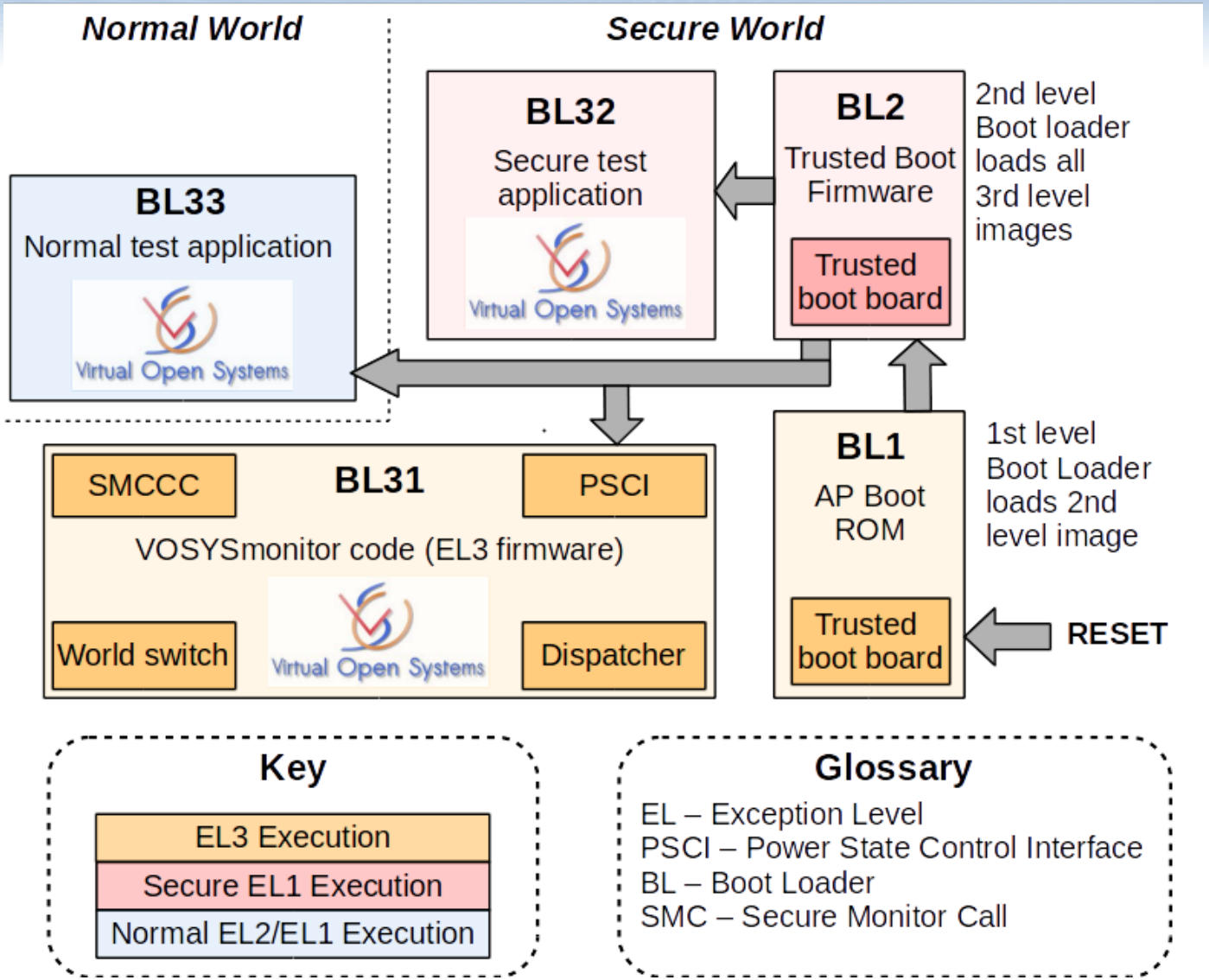
The VOSYSmonitor architecture is split into four parts to ease:

- The support of new hardware platforms and/or Trusted OS
- Software components re-use



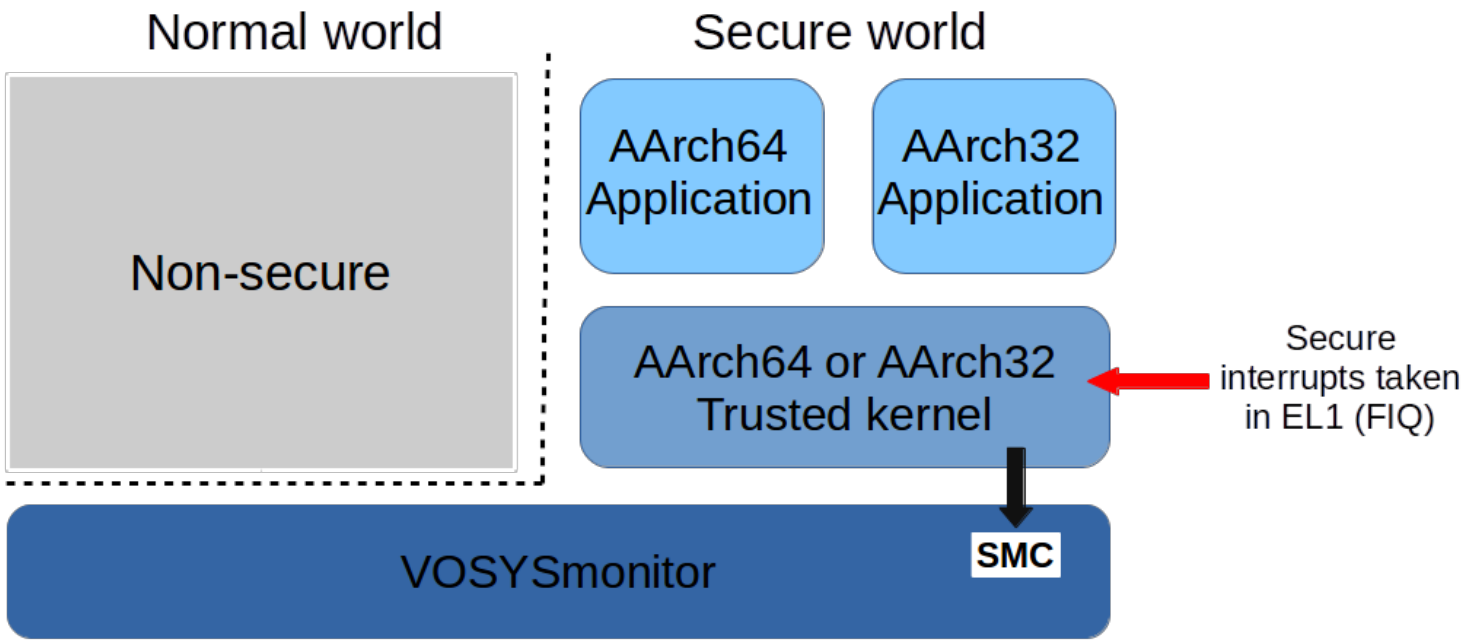
- Drivers to access peripherals requested by Monitor Layer runtime
- Programmable interface between the Secure OS and the monitor layer, running in EL3, to dispatch SMC secure services as well as to handle interrupts forwarding.
- Platform API functions to abstract driver function calls from Monitor Layer
- Monitor layer implemented for ARMv8 architecture, which handles context switching operation and routes trapped exceptions to the right handler

VOSYSmonitor booting order



VOSYSmonitor interaction (1/2)

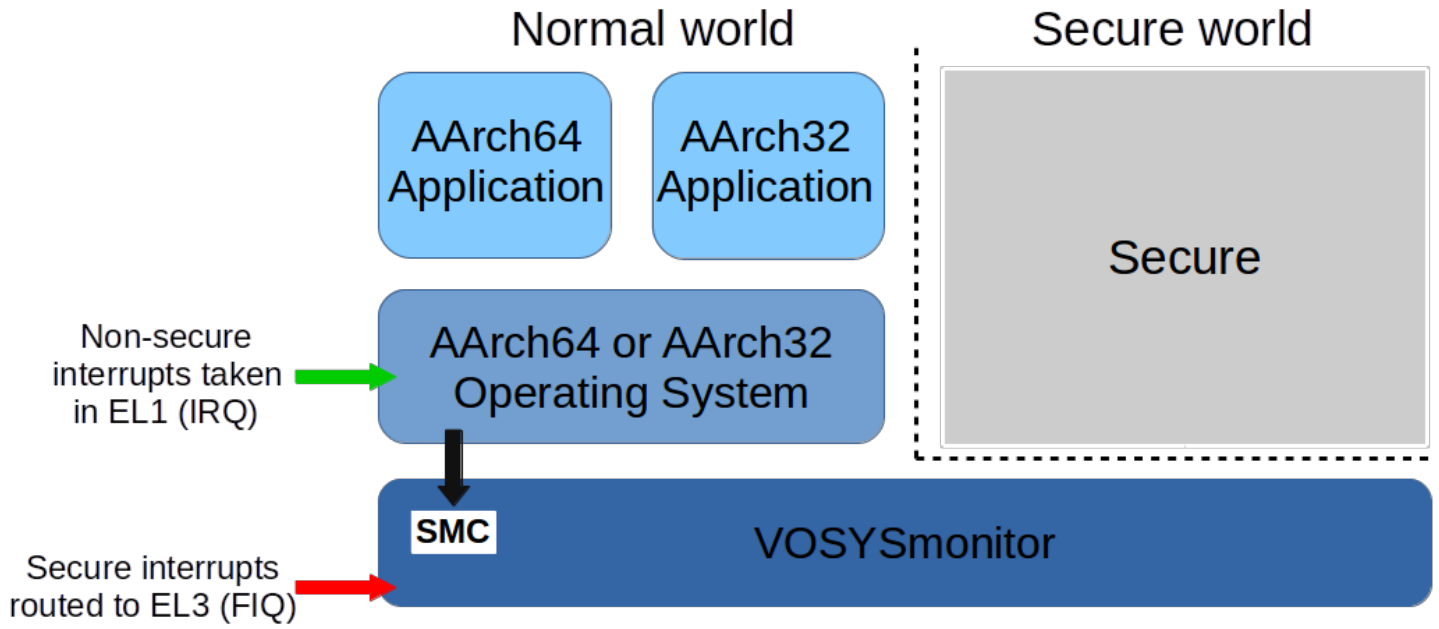
Secure world execution:



- Normal world only executes upon a request by the secure world (SMC)
- FIQ are directly handled in S-EL1 (ensuring low latency)
- IRQ are masked during the Secure world execution.

VOSYSmonitor interaction (2/2)

Normal world execution:



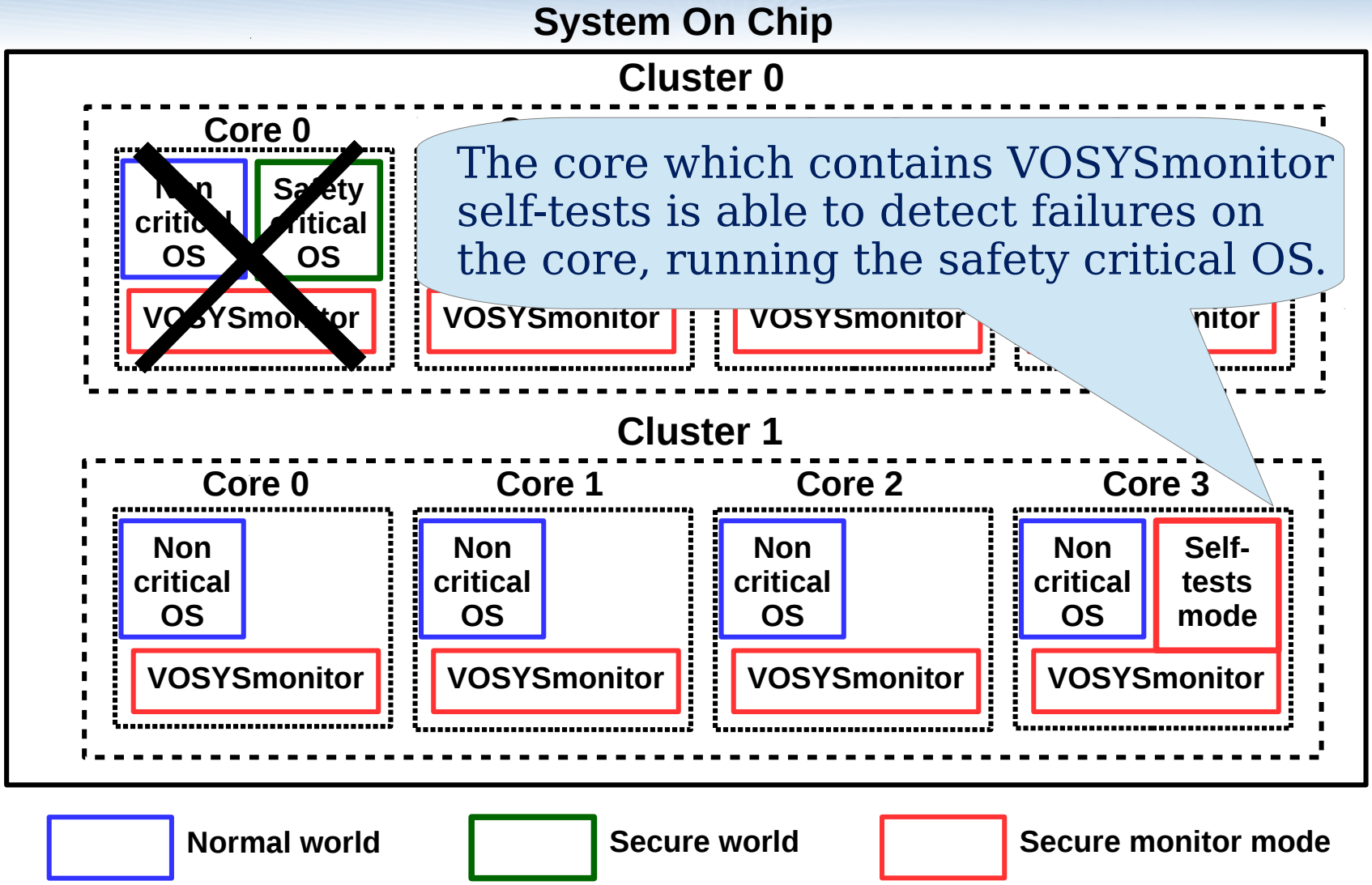
- Normal interrupts (IRQ) are directly handled in NS-EL1
- Secure interrupts (FIQ) are trapped in the Secure monitor firmware to forward it to the Safety critical OS (Normal world preemption)
- Normal world can call secure services through SMC

VOSYSmonitor safety: ASIL-C certification

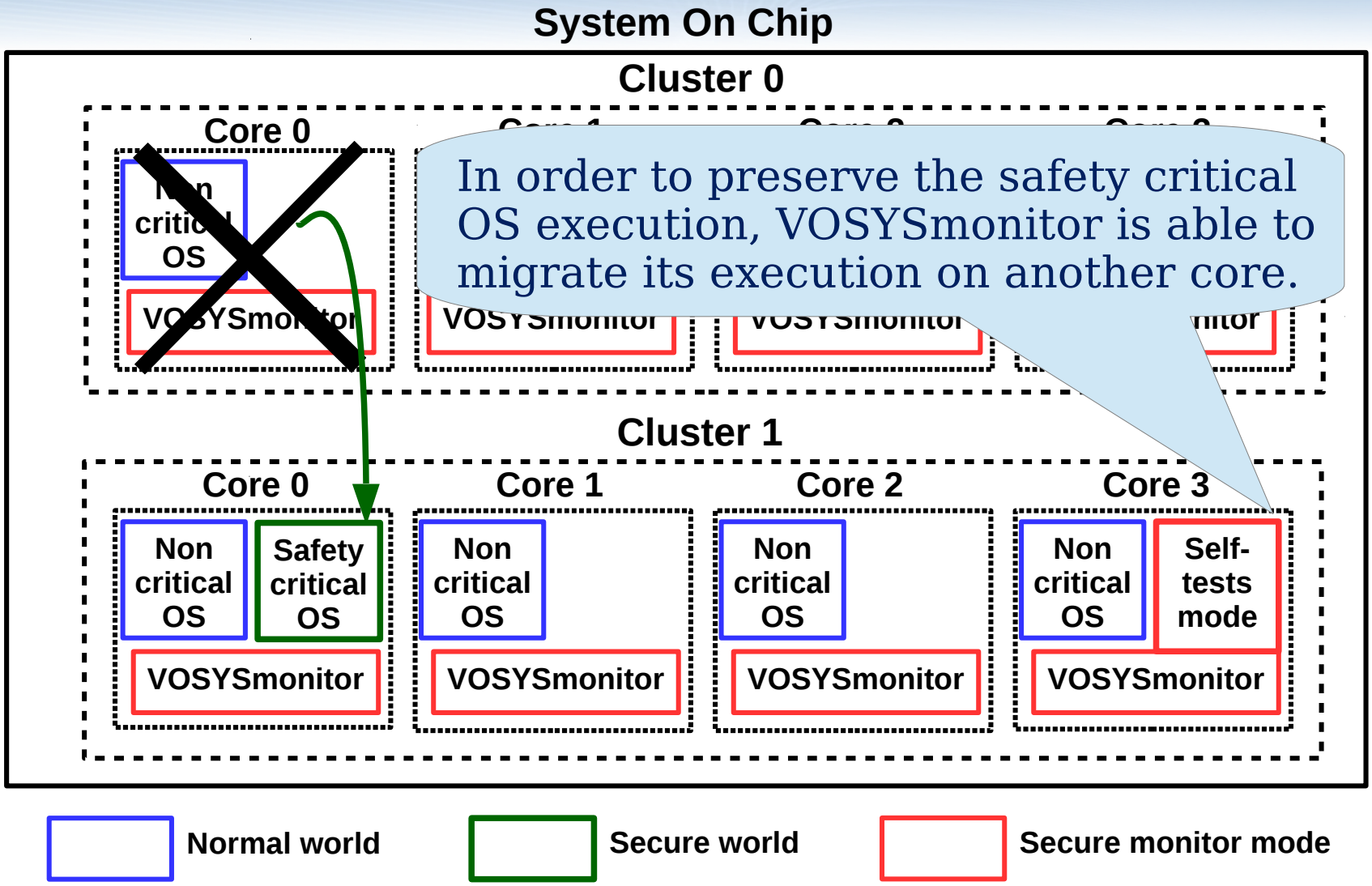
- VOSYSmonitor has been developed with ISO 26262 requirements in mind and as such includes:
  - ***Functional safety package:*** Safety manual, Tool evaluation document, Software verification report, etc
  - ***Self-tests:*** Perform self-tests (memory checking, context switch measurement, etc) during the VOSYSmonitor runtime in order to detect potential failures
  - ***Safe state:*** Preserve the execution of the safety critical OS in case of failures.
- VOSYSmonitor targets ISO-26262 ASIL C compliance.

ISO 26262

VOSYSmonitor safety: Failure detection



VOSYSmonitor safety: RTOS migration



Presentation Plan

- Centralized ECUs integration on ARMv8-A
- VOSYSmonitor for automotive mixed-criticality systems
- **Benchmark and Conclusion**

Performance: VOSYSmonitor setup time

Test case: Use the Performance Monitoring Unit (PMU) to have a very detailed view of latency in terms of clock cycles counter. Start the PMU at the VOSYSmonitor entrypoint and stop it just before jumping to the Secure OS entrypoint.

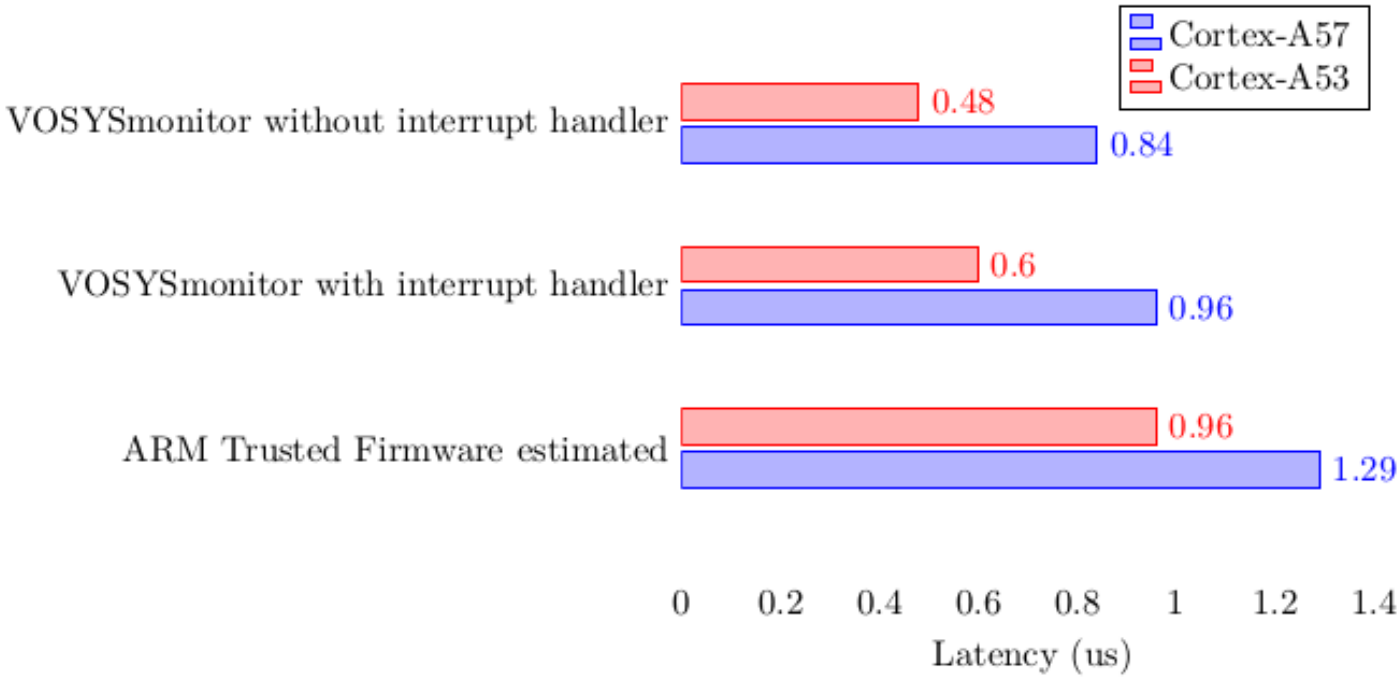
	VOSYSmonitor	ATF
Time (us) JUNO board Frequency 700MHz	17,74 us	853,48 us
Time (us) RCAR-H3 board Frequency 1,3GHz	45,7 us	1119,81 us

VOSYSmonitor setup includes: ARM EL3 initialization, platform peripheral initialization (e.g., Interrupt controller, etc), SMC service, Secure Timer, etc.

Performance: VOSYSmonitor interrupt latency

Test case: Set a timer (free-running mode) interrupt in FreeRTOS. When the FreeRTOS fiq handler is reached, the timer value is compared with the trigger value in order to measure the time consumed before handling the interrupt in FreeRTOS.

Target: Renesas R-Car H3 with 1.5 GHz A57 cluster and 1.2 GHz A53 cluster.



An interrupt framework can be used to distinguish FIQ assigned for the Secure world or VOSYSmonitor, which implies an additional overhead around 100ns.

Conclusion: VOSYSmonitor a flexible software component

Drones

Run a safety software (e.g., landing, overflight, back to home, etc) in case of the main control OS crash.

Public transports

All means of transport, mixed-criticality systems, could be covered by VOSYSmonitor.

VOSYSmonitor

Telecoms

Handle Android OS and secure applications (e.g., online-payment, fingerprint, DRM, etc)

Electrical industry

Circuit breaker: Human Interface Module (HMI) management and overvoltage monitoring

All possible use-cases with mixed-criticality systems.

Virtual Open Systems