# Multi-criticality Hypervisor for Automotive Domain

**CONFERENCE PAPER** · SEPTEMBER 2014



**4 AUTHORS**, INCLUDING:

João Serra
iTGROW ACE, Coimbra, Portugal
**1** PUBLICATION **0** CITATIONS
SEE PROFILE

Tony Almeida
University of Coimbra
**19** PUBLICATIONS **13** CITATIONS
SEE PROFILE

A. Paulo Coimbra
University of Coimbra
**58** PUBLICATIONS **194** CITATIONS
SEE PROFILE

# Multi-criticality Hypervisor for Automotive Domain

João Serra[13], João Rodrigues[1], Tony R. O. Almeida[3], and A. Paulo Coimbra[23]

[1] Critical Software SA, Coimbra, Portugal
jfserra@itgrow.pt, jprodrigues@criticalsoftware.com
[2] Institute of Systems and Robotics, Coimbra, Portugal
acoimbra@deec.uc.pt
[3] Department of Electrical and Computer Engineering – University of Coimbra, Coimbra, Portugal
tony.almeida@uc.pt

**Abstract.** xLuna is a real-time kernel technology that enables concurrent mixed-criticality applications running simultaneously on the same hardware platform, bridging a safety critical application, hard-real-time task set and certifiable real-time operating system alongside a feature rich, non-secure, non-critical, non-real-time general purpose operating system.
The xLuna program, an hypervisor originally developed by Critical Software for space applications, has now a new development focused on the automotive domain extending the principles of its predecessor to new, multi-core hardware. To showcase this new approach, a certifiable real-time operating system is responsible to handle the eCall task, while running simultaneously with the general purpose operating system Android OS on the same hardware platform, by sandboxing Android into a hardware enforced container. Space and time resource constrains are predefined, such as core affinity and static memory allocation, although resource takeover is possible.
This paper will focus on the solutions presented by the hardware used, and how xLuna takes advantage of the technologies it brings to enforce software and resource partitioning.

**Keywords:** Embedded systems, mixed-criticality, multi-core, hypervisor, automotive, infotainment, eCall.

## 1 Introduction

Nowadays, embedded systems are present everywhere. In cars, for example, there are dozens of ECUs (Electronic Control Units) that control one or more electronic safety systems, such as Anti-lock Braking System (ABS) or collision avoidance system, i.e., real-time safety critical systems. Now, with the introduction of in-car multimedia, navigation, web and social capabilities, the number of ECUs tend to increase even further, creating an ever growing need to reduce that number

to minimize production costs, the cars' weight and energy consumption without impacting performance, number of features offered and above all, safety.

Multi-core processors can be a solution for that need by running on a single multi-core ECU what nowadays runs on multiple single-core ECUs. The use of multi-core processors makes possible to reduce the number and weight of electronic hardware (HW), the amount of cabling and, therefore, the weight of the vehicle, making it more efficient and less costly. For instance, let's consider that the core responsible for the eCall application is also responsible for the collision detection and, consequently, to trigger the airbag. In this case, it would be possible to integrate two processing units – one dedicated to the infotainment and the other dedicated to the collision detection – into only one. Furthermore, multi-core processors are an increasing tendency in the world of technology, because they offer a better performance than single-core processors without exacerbating the problems of power dissipation and complexity [1].

Currently, systems with different safety integrity level (e.g., ABS and the Radio) do not coexist at the same ECU. To achieve their integration into a single ECU, it is necessary to isolate systems that would put human life at risk – and as such have a higher level of certification requirements, such as ISO26262 – from systems where a failure would not pose any threat to safety (e.g., the multimedia system). This isolation is usually provided by containing the non-critical system through the use of an hypervisor, allowing both critical and non-critical systems to run together in the same HW platform. Hypervisors are a piece of computer software (SW), firmware or hardware that create and run virtual machines. They are classified into two types: type 1, when running directly on the host's hardware to control the hardware and to manage guest operating systems, and type 2, when running within a conventional operating system environment [2].

This paper presents a new approach of xLuna [3] [4]. This new xLuna is an hypervisor of type 1 with a small footprint that uses ARM® TrustZone® technology [5] to integrate the real-time operating system (RTOS) FreeRTOS™ [6] with the general purpose operating system (OS) Android to run on a Freescale™'s i.MX 6 Quad [7] platform, an ARM Cortex™-A9 [8] based system-on-chip (SoC). FreeRTOS is a free version of the certifiable RTOS SAFERTOS® that provides hard-real-time (HRT) applications and interface with the vehicle networks. This choice was based on the fact that FreeRTOS allows the developers to work on the lowest level of this RTOS, which does not happens with QNX® Neutrino® RTOS [9], the first RTOS considered in this work. Android brings a well-known platform for easier and faster application deployment as well as vendor supplied HW drivers for proprietary non-critical components, such as wireless communications (802.11 a/b/g/n), 3D graphics and sound interfaces, for a richer infotainment experience.

Furthermore, this paper also presents an eCall application that will be futurelly implemented using xLuna. eCall is a pan-European in-vehicle emergency call service based on 112, the common European Emergency number. It was created to improve the time response of emergency teams to car accidents all

over European Union and it will be mandatory in all new cars commercialized in European Union from October 2015 [10]. For more information related to eCall, please refer to section 3.

This work has been partially supported by the ARTEMIS-JU CONCERTO project (n.333053) [11]. This project aims to create a reference multi-domain architectural framework for complex, highly concurrent and multi-core systems. The work developed in xLuna has the objective to implement one of the Run Time Environments for the CONCERTO modelling language.

## 2 Other Virtualization Techniques for Mixed-criticality Systems

The first approach of xLuna [3], developed for space applications, is an hypervisor of type 2. It works on the Real-Time Executive for Multiprocessor Systems (RTEMS) RTOS in order to virtualize Linux within a RTEMS's task. This approach required a lot more SW development and modifications in the OSs' source code than the approach presented in this paper and, due the fact that it runs on a single-core SPARCv8 based processor, it does not take advantage of the multi-core architectures.

There are also others virtualization solutions on the market focused on mixed-criticality multi-core systems for automotive domain. The majority also use certifiable RTOSs but the virtualization techniques used are not specified. On the other hand, Mentor Graphics® and Sierraware created their own hypervisors ready for automotive domain also based on ARM TrustZone technology – respectively, Mentor Embedded Hypervisor [12] and SierraVisor Hypervisor, based on Sierraware's SierraTEE [13].

## 3 eCall System

eCall was developed by the HeERO (Harmonised eCall European Pilot) consortium with the objective to prepare the deployment of the necessary infrastructure in Europe with the aim of making the eCall a reality. The project is partially funded by the European Commission. For more information, please refer to [14]. The eCall enters into scene when a car accident happens, it immediately establishes a 112 emergency call to the nearest emergency centre, Public-Safety Answering Point (PSAP) (Figure 1). It is also transmitted a Minimum-Set-of-Data (MSD) message over IP to that centre containing the exact geographic location of the accident scene, the time of the occurrence, vehicle description, and others eCall details, such as message ID, the trigger mode, etc.

There are three participants in an eCall event – the car, the Mobile Network Operator (MNO) and the PSAP. Each one of them had to be prepared to support the eCall system by following a set of standards. So, xLuna will need to follow the standards dedicated to the cars' manufacturers – CEN EN 15722 (Intelligent transport systems - eSafety - eCall minimum set of data),

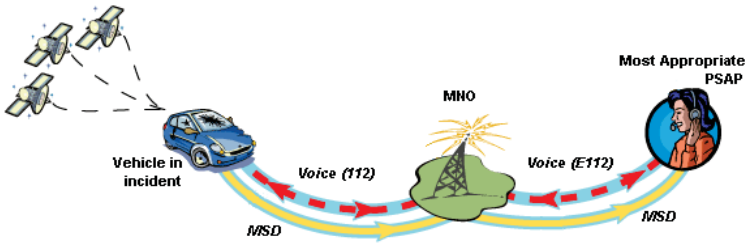



Fig. 1: eCall system [15].

16072 (Intelligent transport systems - eSafety - Pan-European eCall operating requirements) and 16062 (Intelligent transport systems - eSafety - eCall high level application requirements).

## 4 Architecture

TrustZone is present in ARM Cortex-A series processors and it provides the ability to create two environments or worlds, as referred in TrustZone terminology, with different privileges – the secure world and the normal world – providing a new HW assisted software (SW) separation solution, splitting the already existing privileged/unprivileged domains (Figure 2).

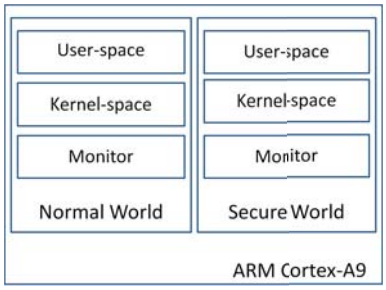



Fig. 2: TrustZone technology architecture.

TrustZone avoids the performance penalty of a world switch every time a CPU exception occurs by having separated exception handler vectors for each world. Otherwise, it would be necessary to call an hypervisor handler that, in turn, would call the correct OS handler, increasing latency as well as higher software development and testing costs. The world switching is done by a CPU exception mode called *monitor*. This mode is entered by calling ARMv7a 'smc' (Secure Monitor Call) instruction. This instruction can be called from either the secure or normal worlds or by a secure IRQ (Interrupt Request)/FIQ (Fast IRQ) that will trigger a world switch.

The TrustZone technology is not limited to the CPU. It also introduces a set of devices that bring support to what was described previously. These devices – TrustZone Address Space Controller (TZASC), Central Security Unit (CSU) and Secure Generic Interrupt Controller (GIC) – are aware of the access requesting world and so they can provide a set of specific functionalities that allows to create a secure container.

The TZASC device enables to configure the system memory up to 16 regions. Each region can be configured as read-only, write-only or read-write to each world, in order to ensure secure memory access. In case of unauthorized access, either read or write, the transaction is denied, a secure IRQ is triggered and a xLuna interrupt handler or interrupt service routine (ISR) is called to handle the situation.

The security enhanced GIC provides the capability to secure interrupts by blocking any configuration attempts from non-secure SW, as well as the possibility to configure a secure interrupt to trigger a FIQ, instead of normal IRQ, that would jump to monitor mode where xLuna would handle it.

The CSU controls the access to bus masters and slaves that are not aware of the currently running world, e.g., the CAN bus interface device, the GPU or the USB. This controller sets which devices are authorized to be accessed by each world and, when an unauthorized access is detected, a secure interrupt is triggered calling a xLuna ISR to give back control to the secure SW.

In addition to the latter devices, the TrustZone Watchdog triggers a secure interrupt if the non-secure SW monopolizes the core for a predetermined amount of time, forcing a switch to xLuna. This time interval, being configurable, allows the HRT scheduler to adjust the amount of computational time it will give to the non-secure environment.

xLuna places itself in the monitor exception routines in order to manage a fast, secure and certifiable entry point. This enables non-critical SW to be sandboxed into this HW enforced container, while ensuring the safety-critical environment has no impact at all. The fact of being an HW implemented container makes the SW required for managing this container substantially less than hypervisors of type 2, making xLuna to be easily converted into a certifiable solution while avoiding any performance penalties of SW emulations.

In order to manage the context switch and to pass the control from one world to the other, a secure Application Programming Interface (API) provides system calls, or monicalls (Monitor Calls) as referred in xLuna terminology, with a specific register set to pass as arguments. Setting these registers followed by the 'smc' instruction will make the current core to jump to the monitor exception handler where xLuna will then securely switch that core from non-secure to secure worlds and call the corresponding handling code to process the requested monicall and arguments.

xLuna also implements an Inter-Core Communication (ICC) mechanism that allows the secure SW to perform a secure request to a target core. It can for example halt the non-secure SW in the target core specified by sending a call to that core.

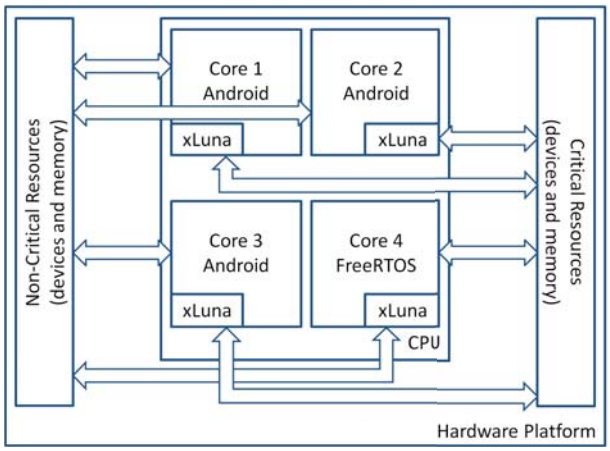


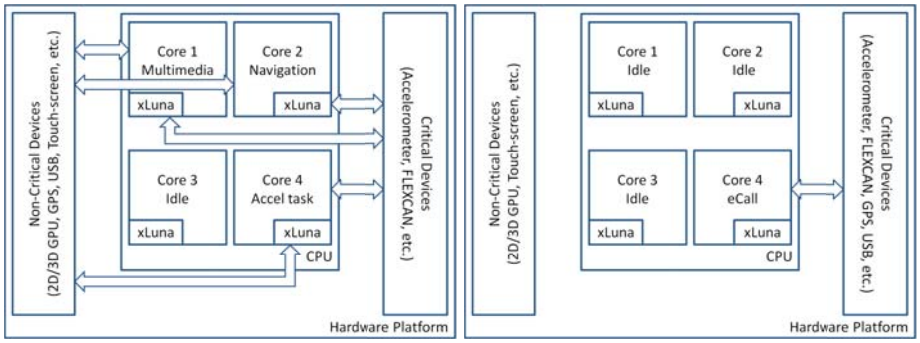

Fig. 3: xLuna hypervisor implementation on a multi-core hardware platform.

As it is shown in Figure 3, in xLuna Automotive, three cores of the processor are dedicated to Android and the only one left is dedicated to FreeRTOS. This attribution was made due to the fact that Android needs more computation resources than FreeRTOS. It can be changed at any time by using xLuna's ICC to halt an Android core and then to start it again, now dedicated to FreeRTOS. Memory is also split into two regions – a larger one, non-critical, for Android, and the other, critical, for FreeRTOS.

## 5 Implementation


Fig. 4: xLuna implementation.

The xLuna for automotive domain has been implemented on an i.MX6Q SABRE for Smart Devices platform with a 10" touch-screen (Figure 4) and devices of interest for automotive critical real-time systems, like FLEXCAN, which implements the interface to the CAN protocol, and an accelerometer. For infotainment, there are 2D/3D GPU, GPS, touch-screen, etc. Secure SW, FreeRTOS, has exclusive access to critical real-time systems related devices. Non-secure SW, Android, has full access to infotainment related devices (Figure 3). The drivers for all the devices needed were developed using C programming language. ARM assembly was used in system initialization code and in exception handling. Linux kernel suffered small changes in UART's device drivers, in order to support the xLuna's monicalls.




(a) Before the eCall event. (b) After the eCall event.

Fig. 5: xLuna takeover example with an eCall event.

In this work, it was accomplish to have Android running simultaneously with FreeRTOS, each one of them with its own tasks running concurrently without interfering with each other.

While Android runs, FreeRTOS has two use-cases tasks. One is responsible for dealing with the USB communication, which is used to connect a 3G modem in order to provide SMS services to the system. The second task is responsible for constantly reading accelerometer values and sending a SMS in case of high acceleration.

With these two tasks, in a demonstrator developed, which showcases a car accident detection, i.e., high acceleration values, xLuna proceeds as follows: (i) it halts Android; and (ii) it sends a SMS to a pre-determined mobile phone number in order to report a car accident and simulate an eCall application.

In the final version of xLuna Automotive, in step (i), in addition of halting Android, xLuna will takeover the non-secure devices needed by eCall application – GPS and USB –, and the step (ii) will be substituted by calling and sending a MSD to 112. In Figure 5, it is illustrated an example of the usage of the CPU

and the devices, as well as the takeover of some of them by xLuna, in an eCall triggered situation.

For this demonstrator (Figure 6), there is an extra task in FreeRTOS that makes a LED to blink to show that the RTOS is alive. When an accident is detected, this LED starts blinking faster.


Fig. 6: xLuna Automotive demonstration prototype.

In the experiments done so far, the results are good. Although neither FreeRTOS nor Android were benchmarked, both OSs presented satisfactory performances. In the case of Android, it was only noticed a small performance degradation for about 10 seconds after its initialization. After that, Android runs smoothly even when a 3D graphics game is running (Figure 7). In the case of FreeRTOS, no performance degradation was noticed.


Fig. 7: 3D graphics game running on Android.

## 6 Conclusion

The usage of TrustZone technology allows to develop an hardware enforced container for separation of software environments with a high-reliably protection against both software and physical attacks. xLuna brings TrustZone to deliver a faster mixed-criticality certifiable solution, enabling feature rich operating systems to coexist with a secure safety-critical real-time operating system, with a small secure paravirtualization layer that has almost unnoticeable performance degradation with no security penalty.

With the monitor call system, new calls can be easily added to expand the underlying secure real-time operating system features to the non-secure environment. For example, it allows access for requesting information through the CAN bus by a secure API to a graphical user interface in the non-secure environment having only to certify what is below that API. This guarantees that there is no impact in the vehicle CAN network, even if the graphical user interface fails for some reason.

The developed system successfully allows to reduce the number of ECUs present in a car that has xLuna installed. Also, it ensures that, as referred above, the real-time operating system accesses the devices needed by the eCall application without any restrictions of time and permissions.

The implementation of the eCall application in xLuna is a relevant novelty. Without being necessary to add new redundant hardware (GPS, accelerometer, communication system, and a processor) to support eCall, a car with the xLuna system will have a feature rich operating system for infotainment, the Android, and, at the same time, the newest safety technology for cars in the European Union.

## 7 Acronyms and symbols

| Abbreviation | Meaning |
|---|---|
| API | Application Programming Interface |
| CAN | Controller Area Network |
| CSU | Central Security Unit |
| ECU | Electronic Control Unit |
| FIQ | Fast Interrupt Request |
| GIC | ARM's Generic Interrupt Controller |
| HRT | Hard-Real-Time |
| HW | Hardware |
| ICC | Inter-Core Communication |
| IRQ | Interrupt Request |
| ISR | Interrupt Service Routine |
| MSD | Minimum-Set-of-Data |
| OS | Operating System |
| RTEMS | Real-Time Executive for Multiprocessor Systems |
| RTOS | Real-Time Operating System |
| SoC | System on Chip |
| SW | Software |
| TZASC | TZ Address Space Controller |

## References


1. Hongtao Zhong, Steven A. Lieberman and Scott A. Mahlke, Extending Multicore Architectures to Exploit Hybrid Parallelism in Single-thread Applications, High Performance Computer Architecture (Scottsdale, 2007), IEEE, 2007, pp. 25–36.
2. Thomas C. Bressoud and Fred B. Schneider, Hypervisor-based Fault Tolerance, ACM Transactions on Computer Systems, ACM, New York, 1996, pp. 80–107.
3. Giovanni Beltrame, Luca Fossati, Marco Zulianello, Pedro Braga and Luis Henriques, xLuna: a Real-Time, Dependable Kernel for Embedded Systems, IP-SOC 2010 Conference (Nov. 30 - Dec. 1), Proc. of the IP Based Electronics System Conference, 2010.
4. xLuna: a Real-Time, Dependable Kernel for Embedded Systems, `http://www.criticalsoftware.com/`, last accessed on June 20, 2014
5. ARM TrustZone: System Security by ARM, `http://www.arm.com/trustzone`, last accessed on June 20, 2014
6. FreeRTOS: a Certifiable RTOS, `http://www.freertos.org/`, last accessed on June 20, 2014
7. Freescale's i.MX 6 Series Multi-core Processors, `http://www.freescale.com/imx6`, last accessed on June 20, 2014
8. ARM Cortex-A Series Applications Processors, `http://www.arm.com/products/processors/cortex-a`, last accessed on June 20, 2014
9. QNX Neutrino RTOS, `www.qnx.com/products/neutrino-rtos/neutrino-rtos.html`, last accessed on June 20, 2014
10. European Parliament: Parliament supports life-saving eCall application in cars, `http://www.europarl.europa.`

eu/news/en/news-room/content/20140224IPR36860/html/Parliament-supports-life-saving-eCall-system-in-cars, last accessed on June 20, 2014
11. The CONCERTO Project, partially funded by the ARTEMIS Joint Undertaking, http://www.concerto-project.org/, last accessed on June 20, 2014
12. The Week In Review: Oct. 25, 2013, Semiconductor Manufacturing and Design Community, http://semimd.com/blog/2013/10/25/the-week-in-review-oct-25-2013/, last accessed on July 20, 2014
13. Shijun Zhao, Qianying Zhang, Guangyao Hu, Yu Qin and Dengguo Feng, Providing Root of Trust for ARM TrustZone using SRAM PUFs, International Association for Cryptologic Research, 2014.
14. HeERO: Harmonised eCall European Pilot, http://www.heero-pilot.eu/view/en/heero.html, last accessed on June 20, 2014
15. European Commission, eCall: Time saved = lives saved, http://ec.europa.eu/digital-agenda/en/ecall-time-saved-lives-saved, last accessed on June 20, 2014