Safety and Security Collaborative Analysis Framework for High-performance Embedded Computing Devices

Irune Yarza[a,*], Irune Agirre[a], Imanol Mugarza[a], Jon Perez Cerrolaza[a]

[a] *Ikerlan Technology Research Centre, Basque Research and Technology Alliance (BRTA), Pº J.M. Arizmendiarrieta, 2, 20500 Arrasate-Mondragón, Spain*

Abstract

Next generation dependable embedded systems are facing a dramatic increase of functionalities and software complexity, requiring heterogeneous high-performance embedded computing devices. The increased software and hardware complexity of such emerging systems, together with the novelty of the technology, poses serious concerns regarding system's safety certification. Moreover, driven by the increased connectivity features, security has become a crucial aspect that shall be considered alongside safety. This article aims to define a safety and security collaborative analysis framework for the systematic analysis of heterogeneous high-performance embedded computing devices. In this direction, a safety and security collaborative analysis methodology is proposed, which is implemented following an incremental top-down strategy. Following this methodology, a generic safety-security model is built first, which is then tailored to a case study for the Zynq UltraScale+ Multi-Processor System-on-Chip (MPSoC) device.

Keywords: safety, security, high-performance, dependable embedded systems, multi-core architecture

1. Introduction

With the increasing digitization trend, dependable embedded systems are moving towards integrated mixed-criticality architectures, where multiple system functions of different safety criticality must co-exist with non-critical software on the same device. Moreover, next generation mixed-criticality systems, integrate advanced features that provide the system with autonomy and higher levels of interconnection. Emerging heterogeneous MPSoC devices are appealing solutions for such systems, providing a rich ecosystem of processing elements that combine multi-cores with Graphics Processing Units (GPUs) and/or Field Programmable Gate Array (FPGA)-based accelerators reaching an unprecedented performance for embedded devices at lower cost and power consumption. However, these benefits come at a cost concerning system dependability (Agirre et al., 2020).

Dependable embedded systems, in which safety is a critical factor (for example an autonomous vehicle), are subject to safety certification. Certification, usually carried out by a third-party, is the determination that a given system is suitable and safe enough for its intended use, where safe enough refers to the absence of unacceptable risks leading to catastrophic consequences caused by the malfunctioning of the embedded system. With the increasing connectivity of embedded control systems, security has become a crucial aspect that shall be considered alongside safety, and security threats shall also be addressed in order to support system safety. However, from a certification point of view, emerging complex MPSoC platform architectures involve many open challenges (Agirre et al., 2020).

Therefore, this article presents a safety and security collaborative analysis framework for heterogeneous high-performance embedded computing devices. Through an incremental top-down safety and

[⋆] The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871465 (UP2DATE) and the Basque Government under grant agreement KK-2019-00035.

[*] Corresponding author

Email addresses: iyarza@ikerlan.es (Irune Yarza), iagirre@ikerlan.es (Irune Agirre), imugarza@ikerlan.es (Imanol Mugarza), jmperez@ikerlan.es (Jon Perez Cerrolaza)

security co-analysis methodology, high level system failures, security threats and their relationships are identified. This analysis is complemented with the definition of both safety and security countermeasures, first for a generic mixed-criticality solution and then tailored to the Zynq UltraScale+ MPSoC platform, where inter-dependencies between safety and security are carefully considered. As a result, the main certification challenges for mixed-criticality integration on high-performance heterogeneous platforms are identified.

The rest of this article is structured as follows. Section 2 provides some background on safety and security co-engineering and mixed-criticality systems. Section 3 provides an overview of safety-security analysis methodologies and heterogeneous high-performance computing platform certification challenges. Section 4 describes the safety and security co-analysis method and an incremental top-down strategy that guides this task. Section 5 describes the generic safety-security analysis for mixed-criticality systems on heterogeneous high-performance computing devices, which is tailored to a case study for the Zynq UltraScale+ MPSoC in Section 6. Finally, Section 7 provides a summary of the article and highlights the conclusions.

2. Background

In this section we provide an overview of functional safety and security standards and mixed-criticality systems.

2.1. Safety & security co-engineering

Functional safety standards, such as, IEC 61508 (IEC, 2010a), ISO 26262 (ISO, 2018), EN 5012x (EN, 2011), define the requirements for the development of safety related electrical and/or electronic systems with the purpose of avoiding unacceptable risks in the system. IEC 61508 is considered the reference safety standard, from which different domain specific standards have been conceived. The assurance level (denoted usually as Safety Integrity Level (SIL)) of the safety functions implemented by the electrical and/or electronic components is assigned based on their severity, frequency of exposure and controllability of the hazardous events. The higher the assurance level, more stringent are the procedures, measures and requirements of functional safety standards.

Several standards are already applied for the design, development, and certification of secure systems, for example the Common Criteria framework (also known as ISO 15408) (Herrmann, 2002), which is mainly focused on IT environments (Mugarza et al., 2017). Due to the increasing number of cyber-attacks against industrial facilities, the International Society of Automation (ISA) developed an industrial cyber-security standard, the ISA/IEC 62443 (IEC, 2010b). Similar to the functional safety IEC 61508 standard, the IEC 62443 introduces the concept of Security Levels (SLs), addressing from casual or coincidental violations to intentional willful attacks using sophisticated means with extended resources, industrial control systems specific skills and high motivation.

Although functional safety and security have historically been treated as two separate disciplines, the International Electromechanical Commission (IEC) has recently published the IEC TR 63069 (IEC, 2019a) and the IEC TR 63074 (IEC, 2019b) technical reports, which provide guidance on the combined application of the IEC 61508 and IEC 62443 standards for the development and maintenance of safe and secure systems.

2.2. Mixed-Criticality

Benefits such as a reduced number of subsystems, devices and associated cabling and connectors that jeopardize the reliability of the system on the traditional federated paradigm, have driven the industry towards the implementation and adoption of integrated mixed-criticality systems where multiple system functions of different criticality are deployed on the same platform. To guarantee the overall safety of mixed-criticality systems, according to IEC 61508 independence shall be achieved and demonstrated at least in the spatial (the data used by a subsystems cannot be altered by another subsystem) and temporal (no element causes another one to miss a deadline due to interferences) domains. Regarding security, in alignment with the IEC 62443 standard, in order to enable the independent security assessment and evaluation of security-related components, a Multiple Independent Levels of Security/Safety (MILS) based solution might be adopted (Heinrich et al., 2019), which enables the integration of components associated to different level of security, using separation mechanisms, such as physical separation or hypervisors.

3. Related Work

Safety and security co-engineering has been an active research area during the last decade resulting in multiple safety and security co-analysis methods that have been surveyed by Lisova et al. (2018); Kriaa et al. (2015). Macher et al. (2015) combines hazard and risk analysis from ISO 26262 (ISO, 2018) for the safety analysis and the STRIDE method (Microsoft Corporation, 2005) for the security analysis. In a similar way, Cui and Sabaliauskaite propose a safety and security co-analysis method where safety and security requirements are first specified and appropriate countermeasures are then selected. The approaches proposed by Schmittner et al. (2014); Steiner and Liggesmeyer (2013) also address safety-security analysis using a Failure Mode and Effects Analysis (FMEA) model and a state or event fault tree respectively to perform a combined safety and security cause-effect analysis. Ponsard et al. (2016) present an integrated safety-security analysis methodology that follows a goal oriented approach to build a goal tree where requirements are connected to related hazards or vulnerabilities and classified as safety or security related.

Besides the need for a safety and security co-engineering, the integration of mixed-criticality systems on complex MPSoC platforms involves many open challenges from a certification point of view. Although cyber-security standards are usually more focused on the software side, functional safety standards include hardware platform related requirements that are not always easily applicable to novel architectures. The stringent and conservative requirements posed by safety standards enforce the application of best industrial safety practices, which leads to relatively slow innovation cycles (Agirre et al., 2017; Rushby, 2008; Martinez et al., 2018). Nowadays a number of multicore safety devices are available in the market, designed mainly for the automotive domain, compliant with IEC 61508 or ISO 26262 standards up to ASIL D, and potentially applicable to other domains (Pérez et al., 2020). However, when these devices are used to integrate mixed-criticality applications, the certification of systems with independence of execution and predictability are still open research challenges that are aggravated with increasing platform complexity (Hassan, 2018). For this reason, in the last decade there have been extensive research in this direction (Pérez et al., 2020; Martinez et al., 2018) and certification experts and standardization bodies have started to include multicore architectures explicitly in standards and guidelines (e.g., automotive domain specific standard ISO 26262 part 11, AUTOSAR's guide for multicore systems (AUTOSAR, 2014; Hassan, 2018), Certification Authorities Software Team (CAST)-32A position paper for multicore processors in avionics (CAST, 2016; Agirre et al., 2017)).

4. Methodology

This section presents a collaborative analysis framework for safety and security that will support and endorse a systematic approach towards safety-security analysis and conclusion extraction.

The proposed methodology is based on the work presented by (Cui et al., 2019; Sabaliauskaite et al., 2016), which is adapted to the objective of this work: a safety-security analysis of mixed-criticality systems running on heterogeneous devices. Moreover, an incremental strategy is proposed, that sets a baseline Safety and Security (S&S) model to be reused as reference in the analysis of different heterogeneous devices.

4.1. Method description

The proposed methodology takes as reference the six step model defined in (Sabaliauskaite et al., 2016), which has also been employed in the collaborative safety and security framework defined in (Cui et al., 2019). The main feature of the methodology is the six-step S&S model (depicted in Figure 1) that analyzes the safety and security of the system through six hierarchies (i.e., functions, structure, failures, attacks and associated safety and security countermeasures) and connects them through relationship matrices to analyze the dependencies among the different elements. The degree of the relationship between the elements at the different hierarchies can be evaluated using different criteria, which can be either qualitative or quantitative (Brissaud et al., 2009). In this report, the qualitative approach, as suggested by the authors in (Brissaud et al., 2009), is used for prior analyzes, where the relationships are graded as very low, low, medium and high impact or coverage. The fulfilling of this analysis on early stages of the development process helps guaranteeing consistency between these six hierarchies (i.e., the effects of a security attack on safety, the failures and attacks

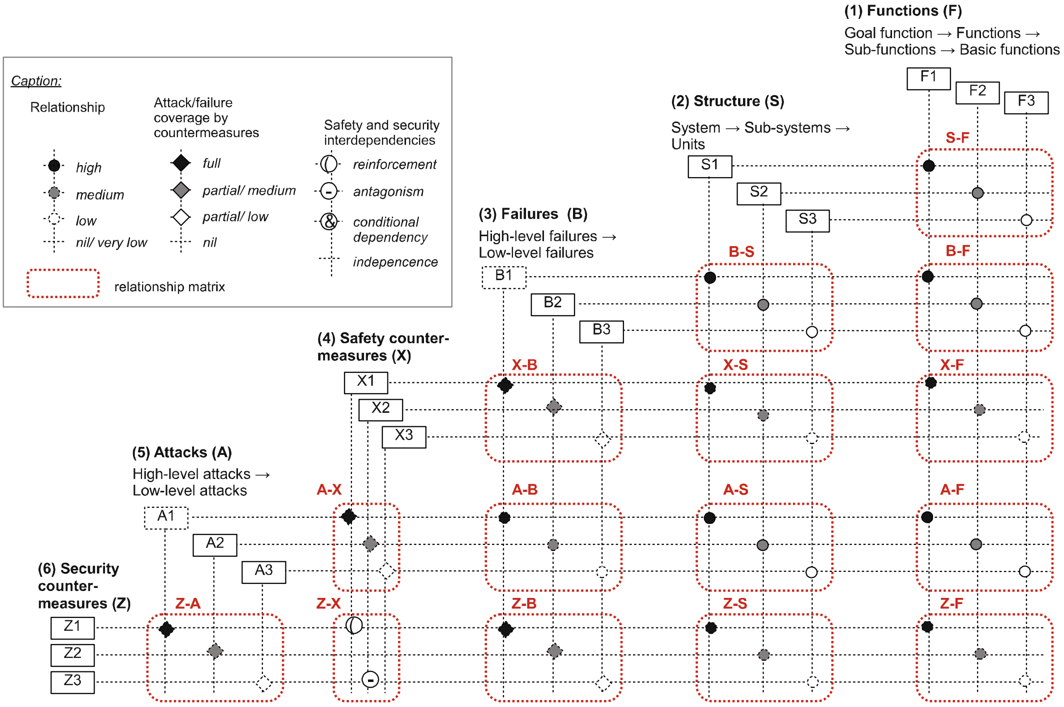

Caption:
Relationship
high
medium
low
nil/ very low
Attack/failure coverage by countermeasures
full
partial/ medium
partial/ low
nil
Safety and security interdependencies
reinforcement
antagonism
conditional dependency
independence
relationship matrix
(1) Functions (F)
Goal function → Functions → Sub-functions → Basic functions
F1
F2
F3
(2) Structure (S)
System → Sub-systems → Units
S1
S2
S3
S-F
(3) Failures (B)
High-level failures → Low-level failures
B1
B2
B3
B-S
B-F
(4) Safety counter-measures (X)
X1
X2
X3
X-B
X-S
X-F
(5) Attacks (A)
High-level attacks → Low-level attacks
A1
A2
A3
A-X
A-B
A-S
A-F
(6) Security counter-measures (Z)
Z1
Z2
Z3
Z-A
Z-X
Z-B
Z-S
Z-F

Figure 1: Integrated Safety and Security analysis model (Cui et al., 2019)

covered by each countermeasure, or even detecting conflicting countermeasures) and understanding the effects of failures and attacks on the system.

4.2. Incremental approach

In the work presented in this article, this methodology is implemented following an incremental approach, which is depicted in Figure 2 and composed of the following 3 steps and S&S models:

1. Generic S&S model: it is based on a generic platform and application patterns. Specifically, generic functions and a typical structure that includes all common components in heterogeneous platforms are considered. In this way, this model defines the key features that shall be analyzed and particularized. The main benefit of this generic S&S model template is its reusability across different implementations, setting a common criteria for the analysis of specific systems.
2. Platform-specific S&S model: taking the previous S&S model as a basis, the system structure is complemented with the particularities of specific heterogeneous platforms. As a result, a platform specific model is obtained which can be reused across different mixed-criticality implementations on such platform.
3. System specific S&S model: the platform-specific S&S model can be taken as a basis to define the system specific S&S model where the functions and their usage of the structure are defined to meet the particular system needs.

This article presents the generic S&S model template in Section 5 and its refinement into a platform specific S&S models for the Zynq UltraScale+ use-case in Section 6. The final step, the system specific S&S model, is out of the scope of this article.

5. S&S Model Template for Mixed-criticality Heterogeneous MPSoC

Based on the methodology described in Section 4, this section builds the generic S&S model template. In order to delimit the scope of the analysis and ease its reusability, this section considers a set of generic functions with different criticality, exemplary assets involved in the security analysis and a typical high-level architecture for heterogeneous high-performance platforms based on an MPSoC (Figure 3). This architecture is taken as the baseline to define high level system failures, attacks and

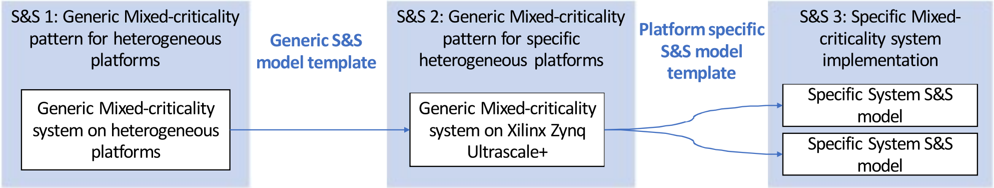

S&S 1: Generic Mixed-criticality pattern for heterogeneous platforms
Generic Mixed-criticality system on heterogeneous platforms
Generic S&S model template
S&S 2: Generic Mixed-criticality pattern for specific heterogeneous platforms
Generic Mixed-criticality system on Xilinx Zynq Ultrascale+
Platform specific S&S model template
S&S 3: Specific Mixed-criticality system implementation
Specific System S&S model
Specific System S&S model

Figure 2: Incremental Safety and Security model development strategy

safety and security countermeasures. In addition, this section provides an analysis of the relationship among these hierarchies with the ultimate goal of evaluating safety and security inter-dependencies.

5.1. Step 1: Generic Functions (F)

Functions of different safety and security implications are considered, building an integrated mixed-criticality system. For S&S model simplicity, a single partition is considered in each category, as this is enough to contemplate the different types of applications and their inter-dependencies. However, as explained in step 3 of the incremental approach (see Section 4.2), this generic S&S model can be then tailored and complemented with the specific system functions and services, which can include various safety partitions with different integrity levels as well as multiple security partitions. Therefore, the *Functions* matrix is comprised of three function categories, although combinations of these function categories (e.g., safety and security-critical applications, or safety-critical and compute-intensive applications) may also exist:

- *F1: Safety-critical applications* ranging from SIL 1 to SIL 4. This application must follow the corresponding certification process for the applicable functional safety standard.
- *F2: Compute-intensive applications* such as, computer vision, image processing, data encryption/decryption or signal processing. Some of these applications require the parallel processing of large data sets. Depending on the application, these functions can also be considered as safety-critical (e.g., autonomous driving) or security critical (e.g., encryption).
- *F3: Security-critical applications* ranging from security level SL 1 to SL 4. These applications, which should fulfill the security requirements of the standard (e.g., IEC 62443) could include an update manager, remote connection, intrusion detection agent, firewall, etc.

5.2. Step 2: Structure (S)

The structure is built based on the components of Figure 3, classified into hardware, hypervisor and OS layers. The hardware considers typical shared resources in modern MPSoC like memory, bus and interconnect, timers, Input/Output (I/O)s, and hardware mechanisms that enforce separation (Pérez et al., 2020). The considered processing units include various multicore Central Processing Unit (CPU) clusters which may be of the same or different architecture, accelerators used for high performance computing workloads (e.g., GPUs or FPGA-based accelerators) and dedicated processors for safety or security purposes. Regarding the software layer, separation kernels with virtualization features can be executed over the different processing units. Given the asymmetric nature of the hardware platform, different hypervisors may coexist in the different processing units of the MPSoC. Over this separation layer different Operating Systems (OSs) are considered, which may have real-time features.

5.2.1. Relationship matrix (SF)

The relationship between structure and functions is based on the resource allocation (Figure 4). As the functions are defined as generic categories, an hypothesis that they all make use of platform resources such as hardware guards, memory, buses, interconnect, timers or I/Os, is made. It is also assumed that all functions depend on an hypervisor or OS. In addition, for the processing units, the following assumptions are made:

- Safety-critical functions run on the multicore CPUs and make use of all platform resources except accelerators.
- Compute intensive functions take advantage of accelerators (e.g., GPU, FPGA) together with one or various CPUs.
- Security-critical functions are allocated to multicore CPUs or security specific modules. They

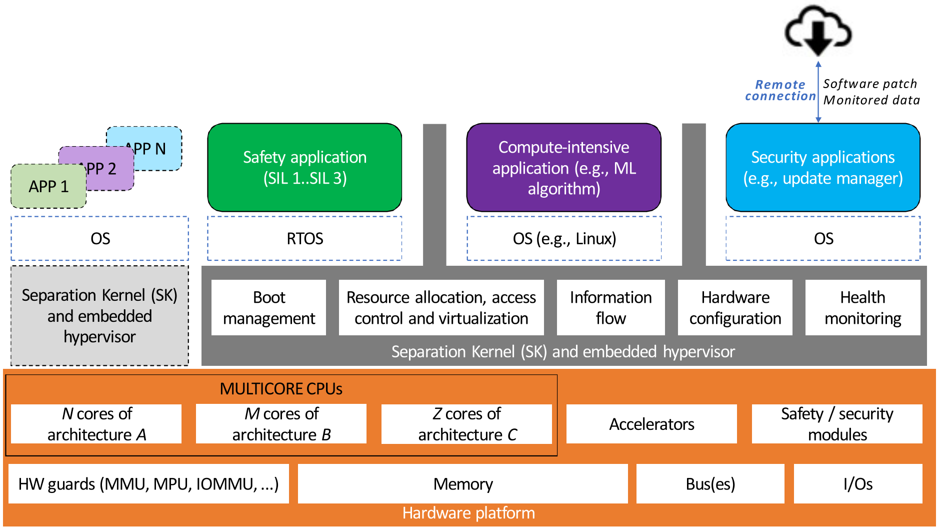
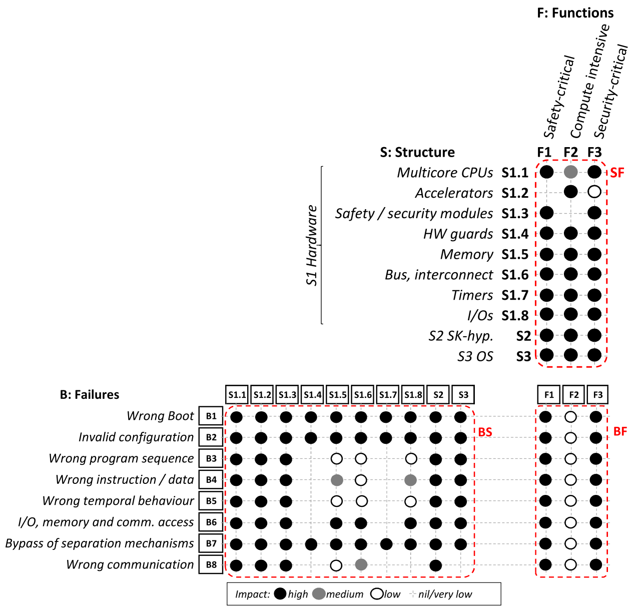

Remote connection
Software patch
Monitored data
APP N
APP 2
APP 1
Safety application (SIL 1..SIL 3)
Compute-intensive application (e.g., ML algorithm)
Security applications (e.g., update manager)
OS
RTOS
OS (e.g., Linux)
OS
Separation Kernel (SK) and embedded hypervisor
Boot management
Resource allocation, access control and virtualization
Information flow
Hardware configuration
Health monitoring
Separation Kernel (SK) and embedded hypervisor
MULTICORE CPUs
N cores of architecture A
M cores of architecture B
Z cores of architecture C
Accelerators
Safety / security modules
HW guards (MMU, MPU, IOMMU, ...)
Memory
Bus(es)
I/Os
Hardware platform

Figure 3: Generic mixed-criticality system on heterogeneous platform



F: Functions

S: Structure		F1 Safety-critical	F2 Compute intensive	F3 Security-critical
S1 Hardware: Multicore CPUs	S1.1	●	◐	●
Accelerators	S1.2		●	○
Safety / security modules	S1.3	●		●
HW guards	S1.4	●	●	●
Memory	S1.5	●	●	●
Bus, interconnect	S1.6	●	●	●
Timers	S1.7	●	●	●
I/Os	S1.8	●	●	●
S2 SK-hyp.	S2	●	●	●
S3 OS	S3	●	●	●

B: Failures		S1.1	S1.2	S1.3	S1.4	S1.5	S1.6	S1.7	S1.8	S2	S3	F1	F2	F3
Wrong Boot	B1	●	●	●	●	●	●	●	●	●	●	●	○	●
Invalid configuration	B2	●	●	●	●	●	●	●	●	●	●	●	○	●
Wrong program sequence	B3	●	●	●		○	○		○	●	●	●	○	●
Wrong instruction / data	B4	●	●	●		◐	○		◐	●	●	●	○	●
Wrong temporal behaviour	B5	●	●	●		○	○		○	●	●	●	○	●
I/O, memory and comm. access	B6	●	●	●		●	●		●	●	●	●	○	●
Bypass of separation mechanisms	B7	●	●	●	●	●	●	●	●	●	●	●	○	●
Wrong communication	B8	●	●	●		○	◐			●		●	○	●

Figure 4: Generic Structure (functions (SF)) and Failures (structure (BS) and functions (BF)) relationship matrices.

could also take advantage of accelerators to offload the CPUs from cryptographic operations.

5.3. Step 3: Failures (B)

The failure modes of an MPSoC could be very extensive. The survey in (Pérez et al., 2020) gives a detailed overview of possible failures and existing mitigation techniques at nanoscale, component and device level. In this section, most relevant device level failures affecting the correct execution of the different software partitions, and hence affecting the mixed-criticality integration on MPSoC platforms, are identified. For this purpose, different component level failures from (Pérez et al., 2020) are taken into account. As a result, eight high-level failure modes are identified (see Figure 4).

5.3.1. Relationship matrices (BS, BF)

The following matrices relate identified high level failures with the platform structure and generic functions. Note that this relationship is based on how system components or functions are affected by a given failure, i.e., the *effects* that a failure can have on the different elements of the structure (and not in the elements involved in the potential cause of failure). For functions, the matrix exposes the severity of the effects in functions, as the template is based on generic functions, the impact of failures cannot be determined. As a result, this phase assumes a high impact for all safety or security critical function categories and a low impact for compute intensive functions that are considered not to be critical and therefore the severity of the consequences of failures should be low. It is then necessary to refine this relationship matrix in the system specific S&S model where the real functions are known.

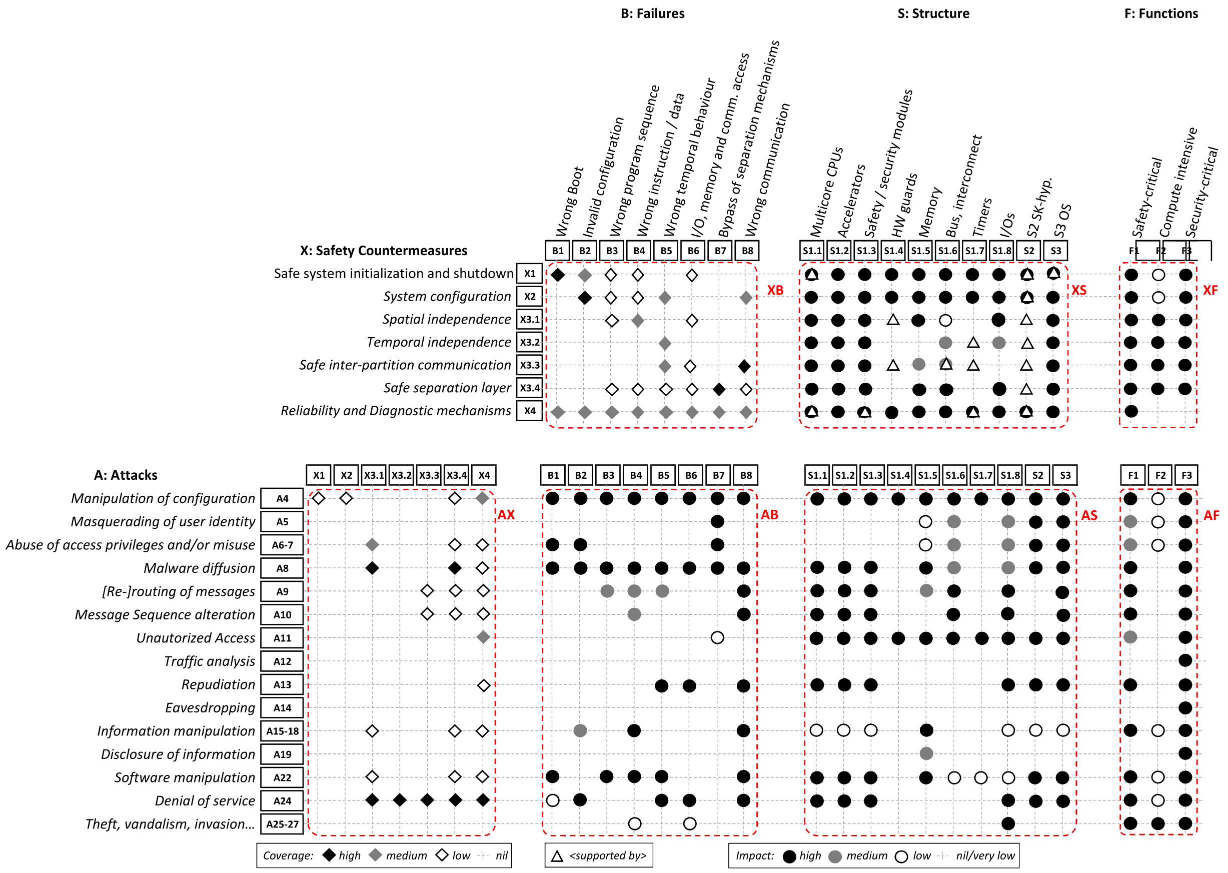

B: Failures
S: Structure
F: Functions
Wrong Boot
Invalid configuration
Wrong program sequence
Wrong instruction / data
Wrong temporal behaviour
I/O, memory and comm. access
Bypass of separation mechanisms
Wrong communication
Multicore CPUs
Accelerators
Safety / security modules
HW guards
Memory
Bus, interconnect
Timers
I/Os
S2 SK-hyp.
S3 OS
Safety-critical
Compute intensive
Security-critical
X: Safety Countermeasures
B1 B2 B3 B4 B5 B6 B7 B8
S1.1 S1.2 S1.3 S1.4 S1.5 S1.6 S1.7 S1.8 S2 S3
F1 F2 F3
Safe system initialization and shutdown X1
System configuration X2
Spatial independence X3.1
Temporal independence X3.2
Safe inter-partition communication X3.3
Safe separation layer X3.4
Reliability and Diagnostic mechanisms X4
XB
XS
XF
A: Attacks
X1 X2 X3.1 X3.2 X3.3 X3.4 X4
Manipulation of configuration A4
Masquerading of user identity A5
Abuse of access privileges and/or misuse A6-7
Malware diffusion A8
[Re-]routing of messages A9
Message Sequence alteration A10
Unatorized Access A11
Traffic analysis A12
Repudiation A13
Eavesdropping A14
Information manipulation A15-18
Disclosure of information A19
Software manipulation A22
Denial of service A24
Theft, vandalism, invasion... A25-27
AX
AB
AS
AF
Coverage: ◆ high ◆ medium ◇ low nil
△ <supported by>
Impact: ● high ● medium ○ low nil/very low

Figure 5: Generic Safety Countermeasure relationship matrices (failures (XB), structure (XS) and functions (XF)) and generic Attack relationship matrices (safety countermeasures (AX), failures (AB), structure (AS) and functions (AF))

5.4. Step 4: Safety Countermeasures (X)

Based on the identified failures, a number of safety countermeasures are added to the model, in such a way that each failure is covered by at least one countermeasure. These countermeasures are also defined as device level mechanisms based on previous work for mixed-criticality integration on multicore platforms (Pérez et al., 2020; Larrucea et al., 2015b,a; Martinez et al., 2018).

5.4.1. Relationship matrices (XB, XS, XF)

Following the methodology, three relationship matrices (see Figure 5) are added to the model:

- **XB matrix** relates safety countermeasures with failures. The matrix specifies the degree of coverage that each countermeasure provides for each failure, with a black symbol for high, gray for medium and white for low. The criteria used for defining the coverage degree is based on the potential failure causes that are covered by the countermeasure. Note that each failure is managed by the combination of various countermeasures. For instance, wrong temporal behavior shall be handled by an appropriate system configuration (X2), temporal independence (X3.2) and safe inter-partition communication (X3.3) implemented with aid of a safe separation layer (X3.4) and diagnostic mechanisms (X4) to achieve the target level of diagnostic coverage according to standards.
- **XS matrix** shows how safety countermeasures affect the components of the system structure. On the one hand, circles specify the components that are protected by countermeasures with a high, medium, or low impact. On the other hand, a triangle is depicted on those components that are necessary or helpful for the implementation of the safety countermeasure. For example, spatial independence has an impact on the processing units, OS, memory and

I/Os as it encapsulates the resources that can be used by the software and it is supported by HW guards, such as the Memory Management Unit (MMU), and the hypervisor that are used to enforce that independence.

- **XF matrix** shows the system functions that are affected by the countermeasures. In general terms, safety-critical functions are expected to be impacted by all countermeasures and all functions are impacted by the countermeasures that guarantee independence of execution (X3.n) among them. Instead, safety related diagnostics only apply to the safety function.

5.5. Step 5: Attacks (A)

This subsection identifies the security threats that may compromise the software updating and monitoring services or the safe operation of the system using the *Magerit* risk analysis methodology (Crespo et al., 2006). The analysis is focused on both intentional and deliberate attacks (see Figure 5).

5.5.1. Relationship matrices (AX, AB, AS, AF)

Considering the identified *Attacks*, four relationship matrices (see Figure 5) are added to the model, as follows:

- **AX matrix** defines which attacks could be covered by safety countermeasures defined in Section 5.4, since some safety countermeasures could be useful to detect possible attacks. For instance, appropriate diagnostic measures could be implemented to detect any casual or coincidental modification in the configuration or the software. Similarly, a watchdog timer or equivalent mechanisms could monitor if a partition has sufficient resources to complete its functions on time, detecting possible denial of service attacks. Additionally, the mechanisms used to guarantee independence of execution could also aid in encapsulating the safety critical partitions against cyber-security attacks. However, as most safety mechanisms are not designed for dealing with intentional and deliberate security attacks, the coverage is generally set as low or medium. Accordingly, this coverage will be complemented with the security countermeasures defined in next section.
- **AB matrix** specifies which of the failures could be caused by a security attack. To aid in the definition of this relationship the potential causes of failures are evaluated.
- **AS matrix** identifies structure elements that could be affected by attacks. In general terms, most attacks associated with services or software could have an impact on the processing units, separation kernel, hypervisor and/or on the OS. Those associated to data or information could also impact the processing units, memory and I/Os. Finally, the attacks related to the communication networks have a higher impact on the bus or interconnect and I/Os.
- **AF matrix** shows the severity of attacks on the different functions. While all attacks are critical from a security point of view, it is assumed that the effect on compute intensive software, considered to be low critical, is low or very low. Safety-critical function instead can be affected with medium to high impact, except for those attacks that only affect information or data confidentiality properties and do not have a direct impact on system behavior.

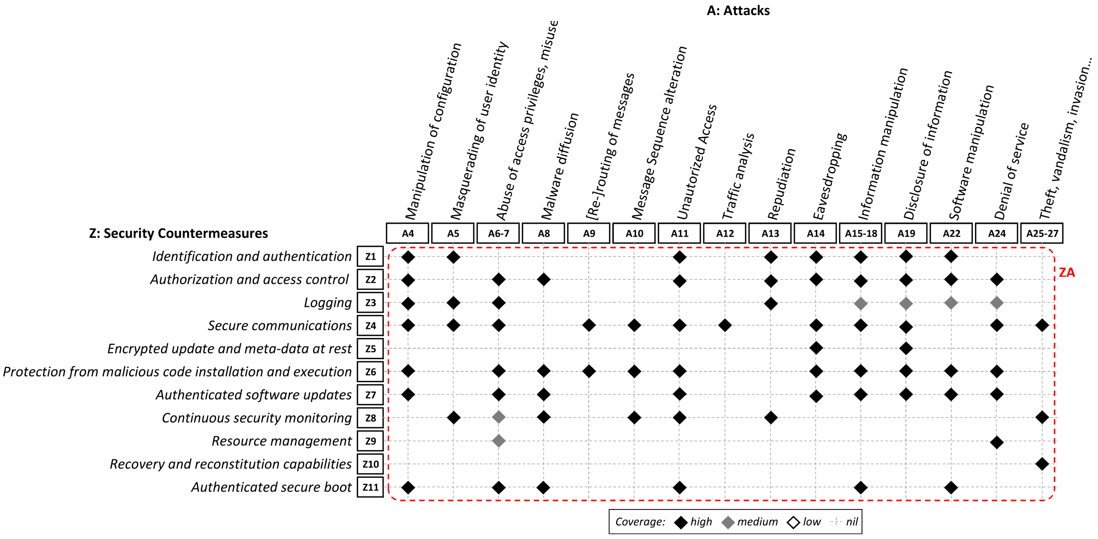
5.6. Step 6: Security Countermeasures (Z)

Aligned with the IEC 62443-4-2 (IEC, 2010b) standard, a set of countermeasures is proposed (see Figures 6 and 7) to mitigate the security threats identified in Section 5.5 (see Figure 5).

5.6.1. Relationship matrices (ZA, ZX, ZB, ZS, ZF)

With the definition of security countermeasures, five new relationship matrices are added to the model (see Figure 6 and Figure 7). From these matrices, the ZX matrix shows one of the most important relationships that should be carefully analyzed to guarantee the alignment and consistency between safety and security.

- **ZA matrix** defines the attacks mitigated by the security countermeasures (see Figure 6).
- **ZX matrix** captures inter-dependencies between safety and security countermeasures. To this end, the three types of relationship defined in (Cui et al., 2019) are used: complement, conflict and independence. In Figure 7 it can be seen how most countermeasures are independent or reinforce the safety of the system. In fact, some security countermeasures could be combined with safety countermeasures to protect against both failures and attacks:



A: Attacks

Z: Security Countermeasures		Manipulation of configuration	Masquerading of user identity	Abuse of access privileges, misuse	Malware diffusion	[Re-]routing of messages	Message Sequence alteration	Unautorized Access	Traffic analysis	Repudiation	Eavesdropping	Information manipulation	Disclosure of information	Software manipulation	Denial of service	Theft, vandalism, invasion...
		A4	A5	A6-7	A8	A9	A10	A11	A12	A13	A14	A15-18	A19	A22	A24	A25-27
Identification and authentication	Z1	◆	◆					◆		◆	◆	◆	◆	◆		
Authorization and access control	Z2	◆		◆	◆			◆		◆	◆	◆	◆	◆	◆	
Logging	Z3	◆	◆	◆						◆		◈	◈	◈	◈	
Secure communications	Z4	◆	◆	◆		◆	◆	◆	◆		◆	◆	◆		◆	◆
Encrypted update and meta-data at rest	Z5										◆		◆			
Protection from malicious code installation and execution	Z6	◆		◆	◆	◆	◆	◆			◆	◆	◆	◆	◆	
Authenticated software updates	Z7	◆		◆	◆			◆			◆	◆	◆	◆	◆	
Continuous security monitoring	Z8		◆	◈	◆		◆	◆		◆						◆
Resource management	Z9			◈											◆	
Recovery and reconstitution capabilities	Z10															◆
Authenticated secure boot	Z11	◆		◆	◆			◆				◆		◆		

Coverage: ◆ high ◈ medium ◇ low ┼ nil

Figure 6: Generic security countermeasure to attack relationship matrix (ZA)

- Authorization and access control (Z2) could be supported with the different privilege levels and restricted read/write permissions established by spatial independence (X3.1).

- The secure communication scheme (Z4) would provide an additional protection layer to all safety-related communications (X3.4) and complement safety diagnostics (X4), ensuring the integrity, confidentiality and authenticity against misuses and intentional or willful cyber-attacks.

- Spatial partitioning (X3.1) can prevent an attacker from installing malicious software (Z6) in other partitions, avoiding propagation.

- The safety diagnostic mechanisms (X4) implemented to detect any independence violation (both in the spatial and temporal domain) might provide additional security continuous monitoring data (Z8).

- Resource management (Z9) is complementary to the mechanisms used to guarantee spatial (X3.1) and temporal independence (X3.2). It also may prevent (or mitigate the effects) of a (distributed) DoS attack.

- The authenticated secure boot (Z11) could be implemented in combination with safe system initialization strategies (X1), by complementing each other.

On the contrary, if special care is not taken, other countermeasures could conflict with each other. For instance:

- The installation of software updates (Z7) could require a reconfiguration of the platform (e.g., the resources required by the latest software patch could be different to the previous software version). This re-configuration may go against the static configuration approach used to guarantee safety (X2).

- Configuration (X2) may be modified also when recovery and reconstitution capabilities (Z10) set the device back into a secure state. In this case, the enforced configuration shall also taken into account safety properties.

- **ZB matrix** identifies those failures that could be mitigated by the security countermeasures. As in the case of attacks handled by safety countermeasures (AX matrix), in most cases the coverage is set as low or medium because security countermeasures have not been originally conceived to deal with these failures. However, as identified in the AB matrix, most failures could be the consequence of an attack (see Figure 5) and therefore, security countermeasures also serve to reduce the probability of such failures. Security countermeasures might also provide an additional system diagnostics capability, mainly focused on software and security (systematic faults). Consequently, the

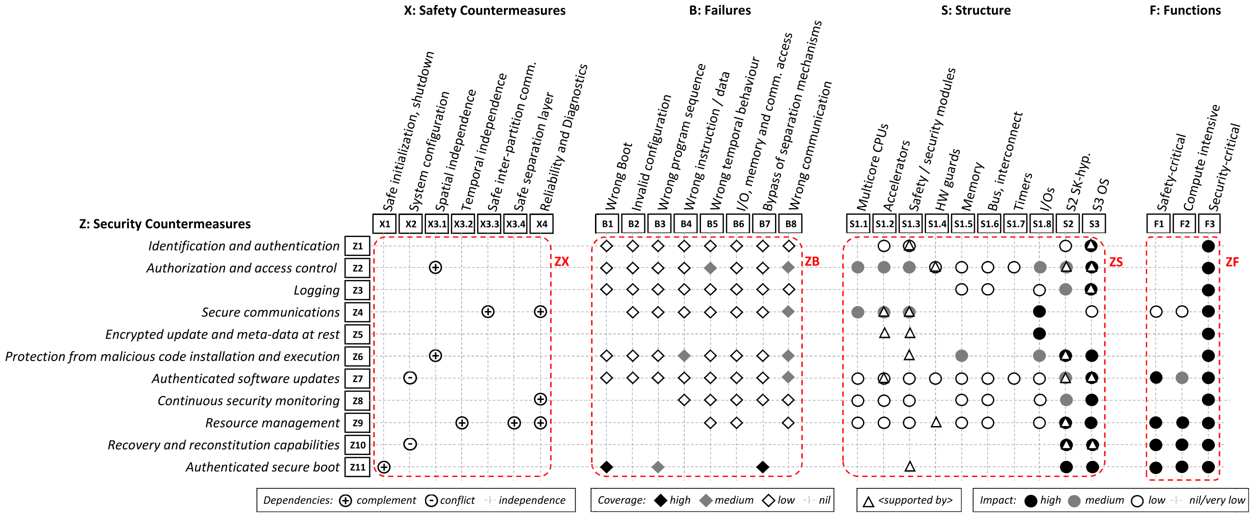

X: Safety Countermeasures
B: Failures
S: Structure
F: Functions
Z: Security Countermeasures
Safe initialization, shutdown
System configuration
Spatial independence
Temporal independence
Safe inter-partition comm.
Safe separation layer
Reliability and Diagnostics
X1
X2
X3.1
X3.2
X3.3
X3.4
X4
Wrong Boot
Invalid configuration
Wrong program sequence
Wrong instruction / data
Wrong temporal behaviour
I/O, memory and comm. access
Bypass of separation mechanisms
Wrong communication
B1
B2
B3
B4
B5
B6
B7
B8
Multicore CPUs
Accelerators
Safety / security modules
HW guards
Memory
Bus, interconnect
Timers
I/Os
S2 SK-hyp.
S3 OS
S1.1
S1.2
S1.3
S1.4
S1.5
S1.6
S1.7
S1.8
S2
S3
Safety-critical
Compute intensive
Security-critical
F1
F2
F3
Identification and authentication
Z1
Authorization and access control
Z2
Logging
Z3
Secure communications
Z4
Encrypted update and meta-data at rest
Z5
Protection from malicious code installation and execution
Z6
Authenticated software updates
Z7
Continuous security monitoring
Z8
Resource management
Z9
Recovery and reconstitution capabilities
Z10
Authenticated secure boot
Z11
ZX
ZB
ZS
ZF
Dependencies: ⊕ complement ⊖ conflict + independence
Coverage: ◆ high ◆ medium ◇ low + nil
△ <supported by>
Impact: ● high ● medium ○ low + nil/very low

Figure 7: Generic security countermeasure relationship matrices (safety countermeasures (ZX), failures (ZB), structure (ZS) and functions (ZF))

ZB matrix is the resulting combination of failures that could be deliberately caused by an attack (AB matrix) and attacks mitigated by security countermeasures (ZA matrix).

- **ZS matrix** shows if the system structure may be affected by security countermeasures. In addition, it uses a white triangle to identify the system structure elements that could be used to support the implementation of the security countermeasures. For instance, identification and authentication countermeasure could be supported by a Hardware Security Module to generate, manage, and store secure cryptographic keys. However, most security countermeasures are usually implemented and enforced at the operating system level and supported by specialized security modules. Some hypervisors, especially those designed for security applications also provide some of the security countermeasures such as access control.

- **ZF matrix** shows how system functions are affected by security countermeasures. Generally, security-critical functions are the responsible of implementing these countermeasures and the ones that show higher impact. However, the implementation of some of the security countermeasures may affect any function like the secure communications or the installation of software updates among others.

6. Case Study: Zynq UltraScale+ MPSoC

Based on the Generic S&S model defined in Section 5, this section presents a platform specific model for the Xilinx Zynq Ultrascale+ MPSoC platform (Xilinx, 2019). The Zynq UltraScale+ platform specific S&S model intends to identify the main safety and security properties (failures, threats and countermeasures) of the MPSoC.

6.1. Structure (S)

Figure 8 defines an example implementation pattern for a mixed-criticality system on the Zynq UltraScale+. The MPSoC consists of a *Processing System (PS)* and a user *Programmable Logic (PL)* section in two isolated power domains within the same device. Within the PS, regions are also defined by individually isolated power domains.

The PS includes two multicore processing units: a *Cortex-A53 Application Processing Unit (APU)*, that according to the chosen family device can either be a quad-core or dual-core MPCore, and a *Cortex-R5 Real-time Processing Unit (RPU)*, a dual-core real-time processing unit that can operate both cores in *lockstep* mode.

The Zynq UltraScale+ integrates also accelerators such as *Mali-400 MP2 GPU*, a graphics processing unit within the PS (not available in the Zynq UltraScale+ CG family), and *Xilinx Deep Learning Processing Unit (DPU)*, a programmable engine dedicated for convolutional neural network that can be integrated in the PL.

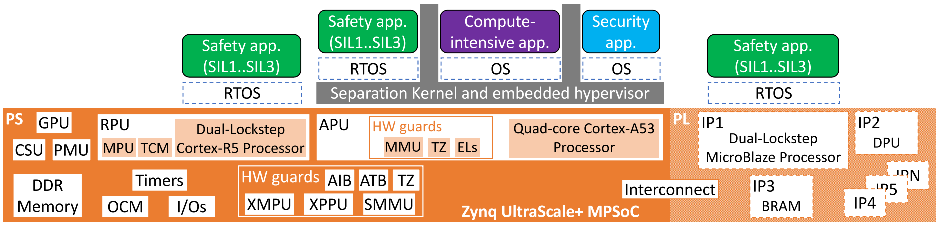


Figure 8: Mixed-criticality system on Zynq UltraScale+ MPSoC

For safety-security purposes, the PS integrates the *Platform Management Unit (PMU)*, a fault-tolerant triple redundant processor that takes care of system initialization, power management and system error handling, and a *Configuration Security Unit (CSU)*, a triple redundant MicroBlaze processor that manages secure boot, on-chip security and temperature monitoring, and supports cryptographic hardware acceleration.

The Zynq UltraScale+ MPSoC offers also multiple hardware guards that provide means to achieve the independence required among partitions, such as *Xilinx Memory Protection Unit (XMPU)* and *Xilinx Peripheral Protection Unit (XPPU)*, *System MMU (SMMU)*, *Advanced eXtensible Interface (AXI) Timeout Block (ATB)* and *AXI Isolation Block (AIB)* and ARM *TrustZone* technology.

The MPSoC integrates also other elements such as system memory, timers, buses or I/O ports. Regarding system memory, the Zynq UltraScael+ consists of a *DDR Memory* and *On-Chip Memory (OCM)* module. Moreover, a *Block Random Access Memory (BRAM)* can also be integrated into the PL as an IP block. The *Interconnect* uses AXI protocol to connect the various system resources within the MPSoC. The Zynq UltraScale+ integrates many different types of timers and counters, such as the *Triple-Timer Counter (TTC)* units in the PS and the various *system watchdog timers* in the interconnect. It offers also a wide variety of I/O ports such as a variety of communication networks, General Purpose I/Os (GPIOs) and safety critical I/O ports.

6.2. Failures (B)

As stated in Section 5.3 the focus of failure analysis is set at device level failures. In this context, shared hardware resources are a potential source of interference and unpredictability, and therefore, also a potential cause of failures. Considering platform specific details and shared resources, the potential cause of some of the generic failures is identified in Table 1. In addition, the effect of those failures is briefly described, focusing on their impact on system structure (marked with a circle in the BS matrix of Figure 9).

6.2.1. Relationship matrix (BS)

The BS matrix in Figure 9 depicts how the Zynq UltraScale+ platform structure components described in Section 6.1 are affected by a given failure. Given that the platform specific S&S model sets the focus on the hardware structure, the BS matrix does not refine the relationship between high level failures and particular separation kernel, hypervisor or OS solutions. Instead, it is considered that the failures could have a high impact in any of them regardless of the chosen solution.

6.3. Safety Countermeasures (X)

The Zynq UltraScale+ MPSoC offers multiple mechanisms (McNeil et al., 2019) that provide means to achieve the independence required between different applications of mixed-criticality. Most generic countermeasures identified in Section 5.4, are applicable to the Zynq UltraScale+ MPSoC, but for X4 the diagnostic particular mechanisms provided by the platform have been identified (see the safety countermeasure matrices in Figure 9).

6.3.1. Relationship matrices (XB, XS)

Considering the platform specific safety countermeasures, two additional relationship matrices (see Figure 9) are added to the model: XB and XS.

The following analyzes the coverage of some of the safety countermeasures (marked in the XB matrix of Figure 9 with a black or gray diamond) and describes how the multiple Zynq UltraScale+ isolation mechanisms, the hypervisor and the OS could assist on the implementation of those safety countermeasures, in other words, it describes the safety

Table 1: Simplified Failure Mode and Effects and Analysis (FMEA) for mixed-criticality partition on Zynq Ultrascale+

Failure Mode	Potential Failure Cause	Failure Effects
B1 Wrong boot & B2 Invalid configuration	Faulty/incorrect boot code in PMU or CSU ROM Faulty/incorrect boot files (memory error or security attack) Faulty/inaccessible Fist Stage Boot Loader Random HW fault at registers, internal memory, HW guard, boot process Modification of configuration by software	High impact in any element of the structure
B4 Wrong instruction/data	Interferences in the shared memories between partitions either on the same processing unit (i.e., APU L2 cache, RPU OCM) or across processing units (i.e., DDR memory, OCM) Random failure in the isolation mechanisms Random HW failures at the processing units (e.g., APU cache, RPU TCM or PMU RAM), the interconnect of any of its submodules	Direct impact on the processing units, the software that runs on them and shared resources and system I/Os
B5 Wrong temporal behavior	Random HW failure in a platform timer/counter (e.g., APU global or core private timer, TTC units or watchdog timers in the interconnect) Faulty interconnect controller (e.g., local PMU and CSU interrupt controllers or RPU, APU and Proxy system interrupt controllers) System integration fault (e.g., scheduling, interconnect arbitration)	Direct impact on the processing units, the software that runs on them and shared resources and system I/Os
B6 Unable to read from or write to an assigned I/O, memory region or other shared resources	Random HW failure in any shared resource or HW guard Random HW failure in the interconnect System integration fault (e.g., scheduling, interconnect arbitration)	Direct impact on the processing units, the software that runs on them and shared resources and system I/Os
B8 Wrong transmission time, data or address in inter-partition communication channel	Random HW failure in any shared resource or HW guard Wrong/unsuitable interconnect arbitration Random HW failure in the interconnect or the communication implemented by the separation kernel/hypervisor or OS	Impact any software running on the processing units using inter-partition communication

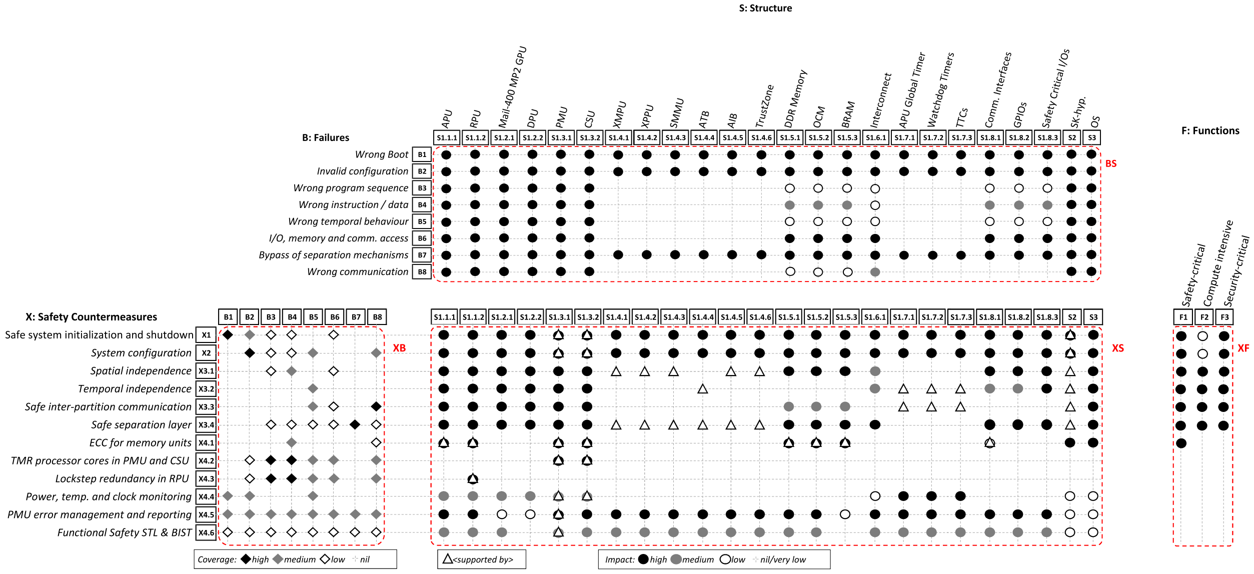

S: Structure
APU
RPU
Mali-400 MP2 GPU
DPU
PMU
CSU
XMPU
XPPU
SMMU
ATB
AIB
TrustZone
DDR Memory
OCM
BRAM
Interconnect
APU Global Timer
Watchdog Timers
TTCs
Comm. Interfaces
GPIOs
Safety Critical I/Os
SK-hyp.
OS
S1.1.1
S1.1.2
S1.2.1
S1.2.2
S1.3.1
S1.3.2
S1.4.1
S1.4.2
S1.4.3
S1.4.4
S1.4.5
S1.4.6
S1.5.1
S1.5.2
S1.5.3
S1.6.1
S1.7.1
S1.7.2
S1.7.3
S1.8.1
S1.8.2
S1.8.3
S2
S3
B: Failures
Wrong Boot B1
Invalid configuration B2
Wrong program sequence B3
Wrong instruction / data B4
Wrong temporal behaviour B5
I/O, memory and comm. access B6
Bypass of separation mechanisms B7
Wrong communication B8
BS
F: Functions
Safety-critical
Compute intensive
Security-critical
F1
F2
F3
X: Safety Countermeasures
B1 B2 B3 B4 B5 B6 B7 B8
Safe system initialization and shutdown X1
System configuration X2
Spatial independence X3.1
Temporal independence X3.2
Safe inter-partition communication X3.3
Safe separation layer X3.4
ECC for memory units X4.1
TMR processor cores in PMU and CSU X4.2
Lockstep redundancy in RPU X4.3
Power, temp. and clock monitoring X4.4
PMU error management and reporting X4.5
Functional Safety STL & BIST X4.6
XB
XS
XF
Coverage: ◆high ◆medium ◇low nil
△<supported by>
Impact: ●high ●medium ○low nil/very low

Figure 9: Zynq UltraScale+ failures (structure (BS)) and safety countermeasure relationship matrices (failures (XB) and structure (XS))

countermeasurestructure relationship marked with a triangle in the XS matrix of Figure 9.

X1 Safe system initialization and shutdown & X2 System configuration. A safe system initialization and shutdown covers a wrong boot or invalid configuration failure among others, whereas the system configuration prevents, an invalid configuration, a wrong temporal behavior and a wrong communication. The boot-up and configuration process supported by the PMU, CSU and hypervisor is highly customizable and can be made secure using ARM TrustZone technology to ensure that boot code runs in the Secure World. In the pre-configuration stage and after a power-on reset, the PMU is responsible for starting the boot process and handling the primary pre-boot tasks (e.g., initialize system monitors, clear PMU Random Access Memory (RAM)), then it releases the CSU from reset. At the configuration stage, the CSU initializes OCM, determines the boot mode through the corresponding register and loads the First Stage Boot Loader (FSBL) code into OCM in both secure and non-secure boot modes. The processing unit that runs the FSBL is configurable, so either the Cortex-A53 APU or Cortex-R5 RPU, can run the FSBL. It is the FSBL that sets-up the user defined isolation configuration and although it is possible to reconfigure at runtime the defined isolation registers, although this is not recommended for safe and secure systems. Therefore, to protect XMPU and XPPU configuration preventing errant or untrusted software from modifying these settings and compromising the intended isolation, the device permits blocking XPPU at runtime and assigning XMPU to the PMU.

X3.1 Spatial independence. The spatial independence helps avoiding a wrong instruction or data failure. To this end, using XMPU, Double Data Rate (DDR) memory and OCM can be partitioned into multiple regions that can be used for communication purposes between the RPU, PMU, and APU subsystems. Regarding peripherals, XPPU uses an ID list to define the processing units that have permission to access each peripheral (i.e., peripherals, message buffers, Inter-Processor Interrupts (IPI), communications, Quad Serial Peripheral Interface (SPI) flash memory). Moreover, the interconnect integrates multiple AIBs that can be configured to achieve independence between the Intellectual Property (IP) blocks in the PL and the various subsystems (i.e. processing units, memory regions, peripherals) in the PS. The interconnect also supports ARM TrustZone isolation, that tags the security level of each AXI transaction.

The Cortex-A53 APU and the Cortex-R5 RPU have a MMU and a Memory Protection Unit (MPU) respectively, which provide means for additional memory access control within each processing unit. The Cortex-R5 MPU supports access protection in L1 cache and external memory. The MMU capability in the Cortex-A53 is extended through the SMMU to Direct Memory Access (DMA) capable devices in the system. Using the hardware virtualization extensions to isolate memory and interrupts as well as the SMMU to isolate DMA capable devices, a hypervisor can set-up multiple virtual environments. The hypervisor can make use of software techniques such as cache coloring to partition the L2 cache shared between the Cortex-A53 CPU cores. The APU also supports four exception levels, providing means for additional access control.

Concerning the FPGA, different spatial isolation mechanisms are supported, such as the route and placement of redundant channels in different silicon die blocks and the definition of isolation fences using specialized qualified tools (see Section 6.3.1).

X3.2 Temporal independence. In order to achieve temporal independence between the various system resources within the MPSoC and avoid a wrong temporal behavior failure, the ATB in the PS interconnect, prevents a master from hanging forever while waiting for the slave response.

Within the Cortex-A53 APU and the Cortex-R5 RPU, it is the responsibility of the system architect to implement a suitable scheduling at software level, which shall take into account possible interferences cause by multicore contention and guarantee that deadlines are met through Worst-Case Execution Time (WCET) analysis. The hypervisor and real-time OS, respectively, shall implement a cyclic scheduling of functional units and ensure that time slots are assigned as statically configured. However, completely mitigating temporal interferences between the different partitions running in parallel and competing for the same shared resources is still an open challenge (Pérez et al., 2020). For this reason, the scheduling table shall be supported by WCET analysis that upper-bounds possible temporal interferences.

X3.3 Safe inter-partition communication. A safe inter-partition communication mechanism prevents from a wrong communication and a wrong temporal behavior among others. To this end, the hypervisors supported by the APU include mechanisms that allow safe data exchange between the various virtual environments. Moreover, the MPSoC supports inter-processor communication using both an IPI interrupt structure and memory buffers to exchange short interrupt-driven messages, with a prearranged communication protocol, between processing units in the system. This is a safe and secure method for two isolated subsystems to exchange information with each other without jeopardizing independence. Access to interrupt registers and message buffers is restricted by XPPU.

X4 Reliability and Diagnostic mechanisms. The RPU can operate both cores in lockstep with physical and temporal diversity and has redundant critical control logic such as lockstep checkers. In order to achieve on chip redundancy with Hardware Fault Tolerance (HFT) 1, it is possible to integrate in the PL an IP of a dual-lockstep MicroBlaze (MB), which executes the same code in strict synchronization. Similarly, the Cortex-R5 RPU together with the dual-lockstep MB processor can be configured to form an on-chip redundant dual-lockstep architecture, which could provide the same fault-tolerant properties as a Triple-Modular Redundant (TMR) solution. Both fail-silent channels can be arranged in lockstep mode, which provides masking capabilities of faults in the CPU as in the TMR approach. Otherwise, they can be used as two parallel fail-silent channels offering double performance capabilities.

Since power domains are physically separated from each other, functional isolation is provided preventing the propagation of failures from one domain to the other. Moreover, design diversity (in hardware) acts as a defense against common-cause development errors. The system monitoring units provide power supply and temperature monitoring capabilities to the MPSoC. Regarding system reset, the PMU provides a flexible reset management, allowing independent reset for the different power domains. Finally, the isolation region or fence defined using Xilinx Isolation Design Flow (IDF) is a structure that fulfills the requirements defined in the Part 2 Annex E Table E.2 of the IEC 61508 and can be implemented to isolate and decouple physical locations within the PL.

Regarding diagnostic, the Zynq UltraScale+ implements particular diagnostic techniques:

- Error-Correction Code (ECC) protection for memory units (e.g., OCM, PMU-RAM, CSU-RAM, and RPU cache and Tightly-Coupled Memory (TCM)), providing a medium coverage against a wrong instruction or data failure.
- PMU and CSU provide high coverage for wrong instruction or data and wrong temporal behavior faults by using TMR processor cores with physical diversity as well as redundant flip-flops for critical control bits (e.g., security state).
- RPU supports running in lockstep mode with physical and temporal diversity and has redundant logic (e.g., redundant lockstep checkers), which provides a high coverage against wrong instruction or data and wrong temporal behavior faults.
- XMPU and XPPU protect memory space from unauthorized system master access.
- Provides multiple watchdog timers in the different power domains to detect systematic and random failures causing program flow errors.
- Provides means for power supply, temperature, and clock frequency monitoring, which might be required in safety applications to cover wrong boot, invalid configuration and wrong temporal behavior failures.
- PMU takes care of system error management and reporting through the system error controller and monitors the activation of common cause failures, providing a medium coverage against every identified failure.
- Includes Built-In Self-Test (BIST), comprising logic BIST for XMPU, lockstep and ECC checkers and memory BIST, to detect hardware faults that may be caused due to a permanent failure.
- The functional safety software test library provides the capability to run tests and evaluate the functionality and status of memory units, Corte-R5 RPU lockstep, PMU, XMPU, XPPU, clocks, voltage and temperature.

6.4. Security Countermeasures (Z)

Security countermeasures are commonly implemented and enforced at the OS-level, although support from specific hardware modules, such as cryptographic engines, is desirable.

6.4.1. Relationship matrix (ZS)

As stated by Xilinx, the Zynq UltraScale+ platform provides fundamental security capabilities for enabling and accelerating the requirements compliance of the industrial cybersecurity standard IEC 62443-4-2 (IEC, 2010b). This section, describes the relationship between the security countermeasures previously identified in Section 5.6 (see Figures 6 and 7) and the Zynq UltraScale+ architecture, matching security hardware modules to the generic security countermeasures.

Z1 Identification and authentication, Z4 Secure communications, Z5 Encrypted update and metadata at rest, Z6 Protection from malicious code installation and execution, Z7 Authenticated software *updates.* The CSU offers an interface block to the internal cryptographic module, providing a rich variety of symmetric and asymmetric cryptographic algorithms (Xilinx, 2019; Peterson, 2018). The cryptographic engine supports the security countermeasures in which cryptographic operations are needed. For instance, the symmetric and asymmetric cryptographic algorithms might be used for the authentication of human users, software processes or other devices. The authenticity of software updates might also be verified through these schemes. On the contrary, as far as the confidentiality and integrity of sensitive data is concerned, the CSU will speed up the data and messages encryption and signing processes required for the secure communications and encrypted data at rest. The provided support can also be used for the protection from malicious code installation and execution. Symmetric and asymmetric key management features are also available, which use battery-backed RAM and electronic FUSEs one-time programming technologies. Optionally, the key loading and storage process can be obfuscated. Key management shall be handled by the software application running on the PS. Moreover, the ARM v8 cryptography extension adds new A32, A64 and T32 instructions for cryptographic acceleration.

Z11 Authenticated secure boot. The CSU provides a secure boot feature (Xilinx, 2019; Peterson, 2018), that verifies the validity of the bitstream and the software image to be executed in the platform. Specifically, the secure boot ensures both the integrity (the software artifacts have not been modified and/or tampered by an attacker) and the authenticity (the software artifacts have been created and released by a trusted party) properties. In addition, the platform offers bitstream and software image confidentiality. This secure boot capability is complementary to the safe system initialization safety countermeasure.

7. Summary and Conclusions

This research work defines a safety and security co-engineering methodology for a systematic safety-security analysis of mixed-criticality systems running on heterogeneous high-performance embedded computing devices. This method enables a comprehensive co-analysis of the system at six hierarchies: functions, its architecture (structure), failures, attacks and associated safety and security countermeasures (Cui et al., 2019; Sabaliauskaite et al., 2016). The defined methodology is implemented following an incremental top-down approach. To this end, the focus is set on high-level system failures and threads and their corresponding countermeasures. First, based on a generic platform and application patterns, a generic model has been built, which has then been tailored to the Xilinx Zynq Ultrascale+ MPSoC platform.

The main benefit of the platform specific model that results form this analysis is its reusability across different implementations, setting a common criteria for the analysis of specific system. Moreover, the fulfilling of this analysis on early stages of the development process helps guaranteeing consistency between hierarchies (i.e., the effects of a security attacks on safety, the failures and attacks covered by each countermeasure, or even detecting conflicting countermeasures). The analysis helps also understanding the effects that failures and attacks may have on the system to select appropriate safety and security countermeasures and analyzing safety and security inter-dependencies. Another relevant aspect of the proposed methodology is related to software updates, since its hierarchical implementation eases the safe and secure integration of software updates during system life-cycle.

This analysis is the first fundamental step in the design of mixed-criticality systems on the selected platform, providing a systematic approach for the development of safety and security concepts. Future work considers refining the defined safety and security countermeasures for specific system implementations, which will also highly depend on the software stack. One of the crucial points will be the mitigation of interference by an in-depth analysis of the architecture not only from manuals, which often contain limited public information, but also by experimentation on the platform.

References

Agirre, I., Abella, J., Azkarate-Askasua, M., Cazorla, F.J., 2017. On the tailoring of CAST-32A certification guidance to real COTS multicore architectures, in: 12th IEEE International Symposium on Industrial Embedded Systems, pp. 1–8. doi:10.1109/SIES.2017.7993376.

Agirre, I., Onaindia, P., Poggi, T., Yarza, I., Cazorla, F.J., Kosmidis, L., Grüttner, K., Abuteir, M., Loewe, J., Orbegozo, J.M., Botta, S., 2020. Up2date: Safe and secure over-the-air software updates on high-performance mixed-criticality systems, in: 23rd Euromicro Conference on Digital System Design (DSD 2020).

AUTOSAR, 2014. Guide to Multi-Core Systems. Report. URL: https://www.autosar.org/fileadmin/user_upload/standards/classic/4-1/AUTOSAR_EXP_MultiCoreGuide.pdf.

Brissaud, F., Barros, A., Brenguer, C., Charpentier, D., 2009. Reliability Study of an Intelligent Transmitter. Proceedings of the 15th ISSAT International Conference on Reliability and Quality in Design URL: https://hal.archives-ouvertes.fr/hal-00430431/.

CAST, 2016. Multi-core Processors - Position Paper.

Crespo, F.L., Gómez, M.A.A., Candau, J., Manas, J., 2006. MAGERIT-version 2 Methodology for Information Systems Risk Analysis and Management Book I-The Method. Ministerio de administraciones públicas .

Cui, J., Sabaliauskaite, G., . US2: An Unified Safety and Security Analysis Method for Autonomous Vehicles, in: Future of Information and Communication Conference, Springer. pp. 600–611. doi:10.1007/978-3-030-03402-3_42.

Cui, J., Sabaliauskaite, G., Liew, L.S., Zhou, F., Zhang, B., 2019. Collaborative Analysis Framework of Safety and Security for Autonomous Vehicles. IEEE Access 7, 148672–148683. doi:10.1109/ACCESS.2019.2946632.

EN, B., 2011. EN 50128:2011 Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems .

Hassan, M., 2018. Heterogeneous MPSoCs for Mixed-Criticality Systems: Challenges and Opportunities. IEEE Design & Test 35, 47–55. doi:10.1109/MDAT.2017.2771447.

Heinrich, M., Vateva-Gurova, T., Arul, T., Katzenbeisser, S., Suri, N., Birkholz, H., Fuchs, A., Krau, C., Zhdanova, M., Kuzhiyelil, D., 2019. Security Requirements Engineering in Safety-Critical Railway Signalling Networks. Security and Communication Networks 2019. doi:10.1155/2019/8348925.

Herrmann, D.S., 2002. Using the Common Criteria for IT security evaluation. Auerbach publications. doi:10.1201/9781420031423.

IEC, 2010a. IEC 61508 Functional safety of Electrical/Electronic/Programmable Electronic safety-related systems (Second edition). International Electrotechnical Commission .

IEC, 2010b. IEC 62443: Industrial communication networks - network and system security.

IEC, 2019a. IEC TR 63069: Industrial-process measurement, control and automation Framework for functional safety and security. 1.0 ed.

IEC, 2019b. IEC TR 63074: Safety of machinery Security aspects related to functional safety of safety-related control systems. 1.0 ed.

ISO, 2018. ISO/DIS 26262 Road vehicles Functional safety (Second edition).

Kriaa, S., Bouissou, M., Pietre-Cambacedes, L., Halgand, Y., 2015. A Survey of Approaches Combining Safety and Security for Industrial Control Systems. Reliability Engineering System Safety 139, 156–178. doi:10.1016/j.ress.2015.02.008.

Larrucea, A., Pérez, J., Agirre, I., Brocal, V., Obermaisser, R., 2015a. A Modular Safety Case for an IEC-61508 Compliant Generic Hypervisor, pp. 571–574. doi:10.1109/DSD.2015.27.

Larrucea, A., Pérez, J., Obermaisser, R., 2015b. A Modular Safety Case for an IEC 61508 Compliant Generic COTS Processor, pp. 1788–1795. doi:10.1109/CIT/IUCC/DASC/PICOM.2015.269.

Lisova, E., Šljivo, I., Čaušević, A., 2018. Safety and Security Co-Analyses: A Systematic Literature Review. IEEE Systems Journal PP, 1–12. doi:10.1109/JSYST.2018.2881017.

Macher, G., Sporer, H., Berlach, R., Armengaud, E., Kreiner, C., 2015. SAHARA: a security-aware hazard and risk analysis method, in: Design, Automation & Test in Europe Conference & Exhibition, IEEE. pp. 621–624. doi:10.7873/DATE.2015.0622.

Martinez, I., Bower, G., Chauvel, F., Haugen, Ø., Heinen, R., Klaes, G., Larrucea Ortube, A., Nicolas, C.F., Onaindia, P., Pankhania, K., Pérez, J., Vasilevski, A., 2018. Safety Certification of Mixed-Criticality Systems. CRC Press. doi:10.1201/9781351117821-10.

McNeil, S., Schillinger, P., Kolarkar, A., 2019. Isolation Methods in Zynq UltraScale+ MPSoCs. Xilinx Application Note .

Microsoft Corporation, 2005. The STRIDE thread model .

Mugarza, I., Parra, J., Jacob, E., 2017. Software Updates in Safety and Security Co-engineering, in: International Conference on Computer Safety, Reliability, and Security, Springer. pp. 199–210. doi:10.1007/978-3-319-66284-8_17.

Pérez, J., Obermaisser, R., Abella, J., Cazorla, F.J., Grüttner, K., Agirre, I., Ahmadian, H., Allende, I., 2020. Multi-Core Devices for Safety-Critical Systems: A Survey. ACM Computing Surveys 53. doi:10.1145/3398665.

Peterson, E., 2018. Developing Tamper-Resistant Designs with Zynq UltraScale+ Devices. Xilinx Application Note .

Ponsard, C., Dallons, G., Massonet, P., 2016. Goal-oriented co-engineering of security and safety requirements in cyber-physical systems, in: International Conference on Computer Safety, Reliability, and Security, Springer. pp. 334–345. doi:10.1007/978-3-319-45480-1_27.

Rushby, J., 2008. Runtime Certification, Springer Berlin Heidelberg. pp. 21–35. doi:10.1007/978-3-540-89247-2_2.

Sabaliauskaite, G., Adepu, S., Mathur, A., 2016. A six-step model for safety and security analysis of cyber-physical systems, in: International Conference on Critical Information Infrastructures Security, Springer. pp. 189–200. doi:10.1007/978-3-319-71368-7_16.

Schmittner, C., Gruber, T., Puschner, P., Schoitsch, E., 2014. Security application of failure mode and effect analysis (FMEA), in: International Conference on Computer Safety, Reliability, and Security, Springer. pp. 310–325. doi:10.1007/978-3-319-10506-2_21.

Steiner, M., Liggesmeyer, P., 2013. Combination of safety and security analysis-finding security problems that threaten the safety of a system URL: http://nbn-resolving.de/urn:nbn:de:hbz:386-kluedo-43604.

Xilinx, 2019. Zynq UltraScale+ Device Technical Reference Manual.