TrustSoC: Light and Efficient Heterogeneous SoC Architecture, Secure-by-design

Raphaële Milan, Lilian Bossuet, Loïc Lagadec, Carlos Andres Lara-Nino, Brice Colombier

TrustSoC: Light and Efficient Heterogeneous SoC Architecture, Secure-by-design

Raphaële Milan[1], Lilian BOSSUET[1], Loïc Lagadec[2],
Carlos Andres LARA-NINO[1], and Brice Colombier[1]
[1] Université Jean Monnet Saint-Étienne, CNRS, Institut d'Optique Graduate School,
Laboratoire Hubert Curien UMR 5516, F-42023,
SAINT-ETIENNE, France.
[2] Lab-STICC, ENSTA Bretagne, Brest, France
raphaele.milan@univ-st-etienne.fr

Abstract

In recent years, heterogeneous SoCs, embedding multiple processor cores and programmable logic, have progressed in terms of complexity and performance. They embed more and more components of different natures. From a security point of view, this leads to an increase of the attack surface exploitable by an attacker. The goals of these attacks are to take control of the system and/or have access to sensitive data. To address this issue, in this article, we propose a novel heterogeneous SoC architecture called TrustSoC, which is secure-by-design. Our proposition presents an innovative way of partitioning the system into worlds to provide the designer with different levels of exclusion for the provision of security. Tiny and distributed hardware security wrappers apply policies and actively monitor the SoC communication bus to enforce these levels of security and prevent any unwanted behavior. TrustSoC is a novel proposition that considers both software and hardware approaches to secure the device. We demonstrate our approach by prototyping the security wrappers as well as their operating rules and show that TrustSoC requires minimal changes while significantly improving the state of the art on secure-by-design architectures.

Keywords: Secure system-on-chip, Hardware IP core protection, Hardware security, Secure architecture

Please cite as:

```
@InProceedings{MBL23,
title = {{TrustSoC: Light and Efficient Heterogeneous SoC Architecture, Secure-by-design}},
author = {Milan, Raphaele and Bossuet, Lilian and Lagadec, Loic and Lara-Nino, Carlos Andres
and Colombier, Brice},
booktitle = {Proceedings of the 2023 Asian Hardware Oriented Security and Trust Symposium
(AsianHOST)},
pages = {1--6},
year = {2023},
publisher = {IEEE},
location = {Tianjin, China},
doi = {10.1109/AsianHOST59942.2023.10409311},
isbn = {979-8-3503-4099-0}}
```

1 Introduction

Heterogeneous System-on-a-Chip (SoC) platforms can be found in multiple domains of application due to their flexibility. They are used from general-purpose applications to critical military applications: high-frequency trading, cloud services, telecommunications, etc. To adapt to a wider range of applications, the number and diversity of components in a SoC are increased. Among the more complex heterogeneous SoCs are the SoC-FPGA, such as the FPGA·SoC Intel Agilex, and the AMD-Xilinx Zynq UltraScale+ MPSoC. This paper focuses on these heterogeneous devices, nonetheless, our work can also be applied to other kinds of SoCs.

With the increase of SoC complexity comes a bigger attack surface for a malicious entity. The SoC communication system is a particularly important point of weakness in the security of the system. The works in [Nas+21] and [Bah+21] demonstrate different approaches for securing the system through the SoC communication bus. Unfortunately, these solutions use a RISC-V processor which is not native to commercial SoC-FPGAs. Indeed, most of these platforms feature an array of ARM processors. This is consistent with current trends, as ARM is the leader in the smartphone market [Fit]. The solutions available in the literature also fail to consider the SoC-FPGA whole and do not provide security protections for all the SoC-FPGA components. This paper addresses these issues by presenting a novel lightweight solution for securing heterogeneous SoC-FPGA based on small, efficient security wrappers. We also propose an innovative way of segregating the SoC-FPGA resources into different worlds using an extension of the ARM TrustZone technology [ARM; AF04]. The proposed solution can be fully integrated into current SoC-FPGAs which feature ARM cores. To estimate the cost of the proposed approach we prototype a SoC architecture called TrustSoC. We use an AMD-Xilinx Zynq-7000 SoC-FPGA as prototyping platform to provide experimental implementation and performance results. Our findings show that TrustSoC has a minimal resources overhead, with only a 6% increase in LUT utilization.

The rest of the paper is organized as follows: Section 2 presents the related work. Section 3 describes the threat model. Section 4 presents TrustSoC: the novel lightweight heterogeneous SoC-FPGA architecture secure-by-design and the costs of the estimated approach with the relevant implementation results. Section 5 presents a discussion and comparison of our approach against related works from the literature. Finally Section 6 concludes the paper.

2 Background

SoC-FPGAs are generally divided into two main components: a processing system (PS) which encapsulates the CPU cores, memory elements, and peripherals, and a programmable logic (PL), *i.e.* the reconfigurable fabric or FPGA. Such systems also integrate other components such as large memories, power management units, communication interfaces, application-specific processors, analog components, etc. The programmable logic allows the user to embed their own hardware accelerators in the form of IPs, post-delivery and potentially at runtime. Communications within a SoC pass through system buses like the Advanced Microcontroller Bus Architecture (AMBA). This technology

is available for ARM cores, and both AMD-Xilinx and Intel SoC-FPGAs employ ARM cores. The Advanced eXtensible Interface bus (AXI) [ARM03] is the main communication channel between the processing system and the programmable logic in the case of AMD-Xilinx SoC-FPGAs. In these platforms, a proprietary IP (the AXI interconnect) acts as a translator between the ARM AXI and AMD-Xilinx' own specification: the AXI4 bus.

Given how SoCs may be employed to handle sensitive data, they have become prime targets for malicious entities. The main purposes of attacks on SoCs range from stealing sensitive data to mount a denial of service. These attacks are mainly software-based and target the PS. They are possible, in part, because some SoC resources are shared between applications. As an example, in some recent multi-processor SoC architectures the last level of cache is shared between the different cores. A malware can leverage this characteristic to determine, according to the cache memory access time, whether the target application has accessed the data [OST05]. This provides an adversary with helpful information regarding the target application. This attack is also viable on SoC-FPGAs [BB21].

To secure the execution of critical software applications in modern SoCs, it is common to use built-in solutions such as the ARM TrustZone technology which is available for SoCs with ARM processors [ARM]. This strategy splits the resources of a processor into two different worlds: the secure world and the non-secure world. For SoC-FPGAs, AMD-Xilinx has developed a TrustZone Extension which also protects some of the operations in the programmable logic [AMD14]. Fig. 1 illustrates a didactic example of this technology applied to a SoC-FPGA: the red blocks represent the non-secure world and the green blocks the secure one. To determine in which world it is currently operating, the system relies on the value of the non-secure bit (NS). This protection strategy is applied to the processing system (each CPU core can execute software applications in one of the two worlds), to the memory resources, and also to the programmable logic (each IP embedded in the programmable logic is linked to one of the two worlds). The NS bit is sent through an AXI4 bus to allow the programmable logic to be aware about the world (secure/non-secure) in which the software application is running at any time. This prevents non-secure resources (in the processing system, programmable logic and memories) from accessing secure ones. The code and data within the secure world are supposed to be protected from intruders. It was originally conceived as an efficient, holistic security approach.

Unfortunately, recent works have shown that despite of the protections brought by this security solution, many vulnerabilities can be exploited to perform effective attacks and corrupt the security partitioning. In [BBA19], the authors target the communication system of the SoC. They show that a hardware Trojan can modify the AXI communication signals and force an arbitrary value for the NS bit. This modification can jeopardize the rest of the system, leading to privilege escalation or denial-of-service attacks. In addition to this attack, the work proposed in [BB18] uses the power management of the SoC-FPGA to perform covert transmission of data between secure and non-secure worlds despite the policies of the TrustZone isolation. In [Gro+20], the authors used a hardware Trojan to corrupt the secure boot and break the memory isolation. The modification of the secure boot can allow an attacker to be able to change permissions to critical information, data or instructions, and can lead to

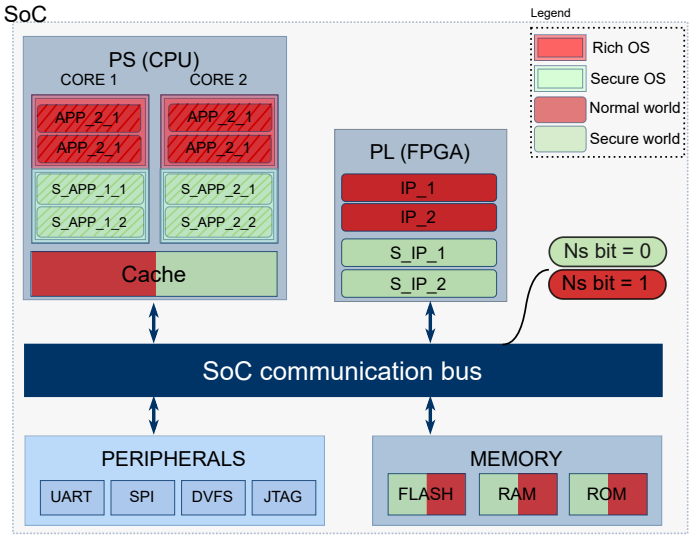


Figure 1: An example of heterogeneous SoC architecture with ARM-TrustZone technology. The red blocks represent the non-secure world and the green blocks belong to the secure world.

privilege escalation.

All these attacks show that it is not sufficient to consider the security of the SoC from just the software or hardware point of view separately. Instead, the design of protections and countermeasures requires an integral approach. Security solutions must be carefully thought out and consider both factors: software (processing system, operating system, boot, etc.) and hardware (programmable logic, bus, hardware IP, etc.).

3 Threat model

In this paper, we contemplate several threats from remote software and hardware attacks. TrustSoC considers malicious hardware IPs or software applications introduced at design time. TrustSoC also considers illegitimate accesses and modifications of the memory contents.

Time-to-market tends to become narrower, designers do not have the time to develop every software or hardware component, thus they utilize third-party blocks. These components can contain malicious routines or circuits which can be used to perform an attack. For example, they could try to access sensitive information from other applications or IPs. These threats we envisage are relevant and correspond to the process of SoC-FPGA design. TrustSoC mitigates these menaces by introducing minimal additional components for every hardware IPs enforcing policies to prevent illegal accesses.

We assume that the software compiler and the synthesis tool are trusted and cannot be used to perform illegal modifications of the design. The synthesis tool is responsible for the additional components addition of every hardware IPs. We also assume that the SoC and the founder are trusted. No modification was

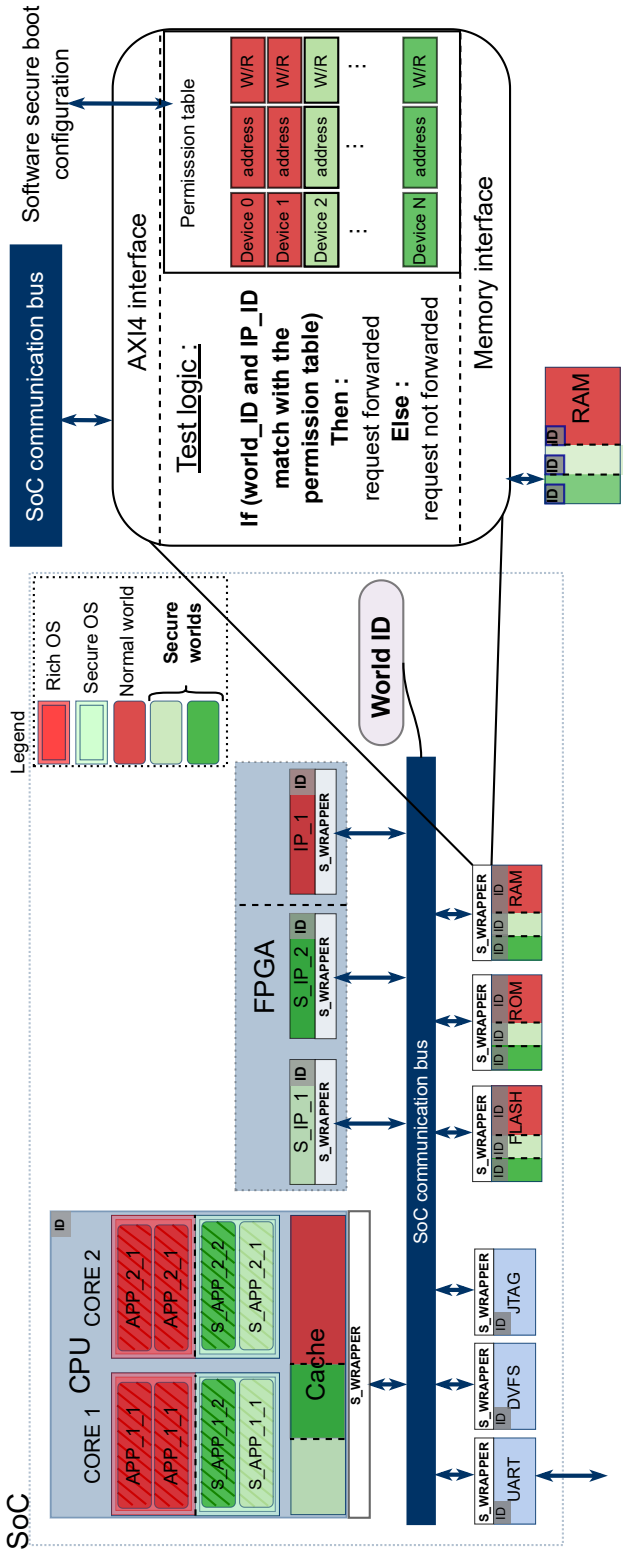


Figure 2: Example of the proposed TrustSoC architecture

made such as adding a hardware Trojan.

TrustSoC prevents the attacks cited in the background Section 2. TrustSoC prevents side-channel attacks [BB21] against the cache memory with several security solutions. TrustSoC uses identifiers to restrict the cache access and creates different cache partitions for each running application. It also stipulates operating rules such as flushing the cache on each context switch preventing reuse of data. Additionally, TrustSoC prevents attacks performed with illegal accesses [Gro+20]. TrustSoC uses additional components that can distinguish between legitimate and illegal accesses with a set of rules and identifiers. With this solution TrustSoC prevents [BBA19].

4 TrustSoC

This section describes an instance of the trusted heterogeneous SoC architecture secure-by-design called TrustSoC. This design exhibits multiple security features:

- software and hardware components can be assigned to multiple worlds with different privilege levels, in contrast to the basic secure/non-secure approach of TrustZone;
- the cache memory is protected against attacks that leverage the shared cache access;
- a set of distributed communication controllers enforce policies to implement trusted communications inside the SoC.

TrustSoC is a flexible and scalable architecture which can be adjusted to the application requirements. For the purposes of this paper we present the concrete prototype shown in Fig.2. The processing system includes two cores, but it could be extended with more cores and support different architectures. TrustSoC also embeds a programmable logic region, a communication bus, several peripherals and shared memories. Each core has a non-secure world, shown in dark red, and two secure worlds, shown in green colors, which are presented in the next subsections. Finally, TrustSoC embeds tiny distributed communication controllers called *"s_wrapper"*. We also present these controllers in the following subsections.

4.1 TrustSoC security features

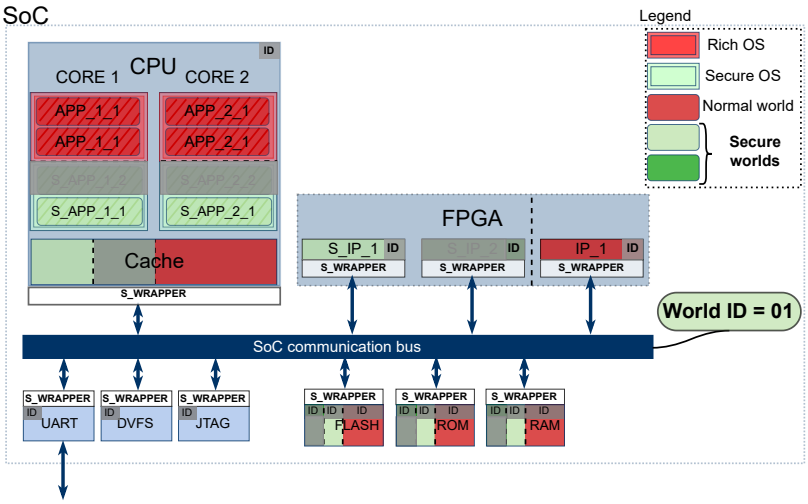
SF.1: Operating rules: TrustSoC comes with a set of operating rules which must be enforced as policies to prevent any unwanted behavior and provide more security.

SF.2: Extended secure multi-worlds: TrustSoC introduces multi-secure domains to allow the designer more flexibility for their design. Contrary to ARM TrustZone technology, TrustSoC allows the designer to choose the number of secure worlds in their design and have a proper isolation between the applications and IPs.

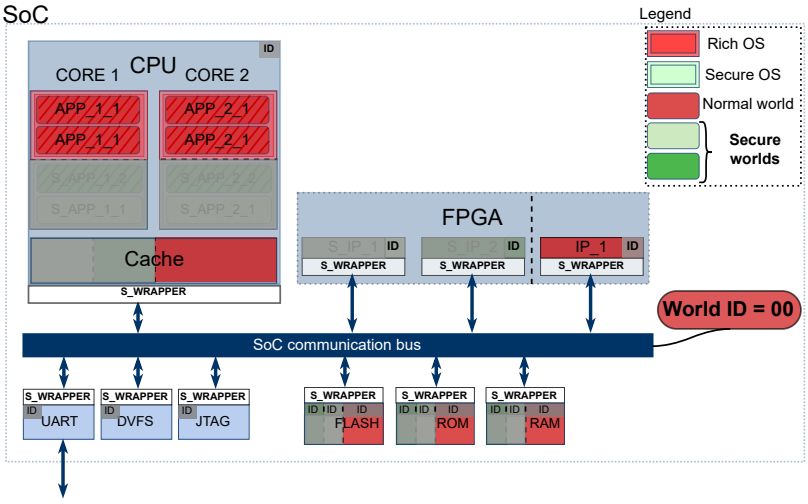
SF.3: Programmable logic in the security resources : TrustSoC fully integrates the programmable logic in the security resources by using a unique identifier for each hardware IP.

SF.4: Trusted communications inside the SoC: TrustSoC establishes secure communications between hardware IPs and software applications inside the heterogeneous SoC. With this security functionality, TrustSoC does not rely on third party's security features and ensures that the IPs introduced are operating as intended.

SF.5: Side-channel attack resilience: TrustSoC embeds protections against software side-channel attacks by restricting the cache access with identifiers and creating different and isolated cache partitions for each running application. The operating rules of TrustSoC also stipulated the flush of the cache on each context switch or end of utilization of the cache by an application.



(a) Operation of one of the secure world



(b) Operation of the non-secure world

Figure 3: Architecture of the operation of TrustSoC

4.2 TrustSoC architecture

TrustSoC embeds ARM Cortex processors. This choice is motivated by the fact that ARM has a strong presence in the SoC and SoC-FPGA markets. ARM processors can be found in the main SoC-FPGAs of AMD-Xilinx and Intel. Nevertheless, it would be possible to employ any other kind of processors in TrustSoC such as RISC-V.

TrustSoC uses small distributed security wrappers to create trusted communications between the hardware accelerators, peripherals and applications. The security wrappers aim to distinguish between illegal and legitimate transactions. To establish this secure communication, TrustSoC assigns an IP identifier and a world identifier to each hardware resource in the SoC. These identifiers are different, unique and hardware-coded. They are assigned pre-synthesis and cannot be changed. Each security wrapper comes with a set of permissions that specifies the access rights for every hardware resource to the underlying component.

These identifiers are transported through the communication bus, which is an AXI instance for the prototype presented in this work. AXI is a slave/master protocol. It has five separate channels: write address (AWADDR), write data (WDATA), write response (BRESP), read address (ARADDR), read data (RDATA) and the optional read response (RRESP). The AXI protocol operates on handshake mechanisms with *ready* and *valid* signals for each channel. The response channels (BRESP and RRESP) indicates the state of the transaction to the master: OKAY when the transaction was successfull, SLVERR or DECERR when an error had occurred.

In addition, the AXI protocol allows to use user signals to transport added information up to 1024 bits without overhead. We leverage this feature in TrustSoC: each request on the SoC communication bus has the IP identifier and the world identifier added through the AXI user signals. The width of the identifiers depends on the number of components and worlds in the SoC. Similarly, for the identifier of the worlds we use $\lceil \log_2(\max(\text{worlds})) \rceil$ bits. The world identifier is used to extended the NS bit of the ARM TrustZone. Since it is hardware-coded and we assume that the AMD-Xilinx tool chain is trusted the world ID cannot be changed preventing attacks like [BBA19]. Additionally, the distributed controller prevents unauthorized modifications. When a security wrapper receives a request, it compares the IP and world identifiers with its list of access policies (read/write). The access rights are hardware-coded but may be changed at boot time through a software secure configuration. After the boot configuration, the policies are set and can no longer be changed. If a request conveys a correct address and world, plus it respects the security policies, it is forwarded to the underlying component. In the case where an anomaly is detected, the wrapper discards the data, sends a null response, and raises an error on the AXI bus through the response XRESP signals. These hardware-coded identifiers and access rights verify the **SF.4** security feature in subsection 4.1. Furthermore, the secure boot configuration that can change the access rights will provide more flexibility to the designer.

The distributed security wrappers also embed simple security policies to oversee the operation of the IP. For example, we specify the reset after every use of the component to prevent the reuse of data. These security policies correspond to the **SF.1** and **SF.3** security features.

4.3 Multiple security worlds

We use an ARM processor in TrustSoC due to its large availability. Our proposal is an extension of the ARM TrustZone technology. We aim to give the designer more flexibility, but especially more isolation on their design since the designer can isolate or group hardware IPs or software applications by worlds. With our trusted communication system and operating rules we address the vulnerabilities of the basic ARM TrustZone technology. We provide isolation between the worlds making it impossible for a malicious entity to get information on a victim that would reside in one of the secure worlds. A potential attacker also could not modify the identifiers to perform an illegal access to a world where it does not belong. Dedicated security wrappers control the accesses to the memories making it also impossible for a malicious entity to perform illegal memory accesses. With TrustSoC, the designer chooses the number of worlds they wish in their system. The encoding of the identifier of the worlds is given by $\lceil \log_2(\max(\text{components})) \rceil$ bits. This enforces the **SF.2** security feature.

The Fig.3 illustrates an example of TrustSoC applied to a design. The secure worlds are identified with different levels of green and the normal world is identified with dark red. This figure shows an example of the encoding in a system where there are three worlds. Fig.3a illustrates the isolation between entities from different worlds: resources of the secure worlds, encoded with world ID = "01", are inaccessible to the normal world in Fig.3b. The hardware and software components in the system cannot communicate directly with each other without authorization. Also, the components cannot access or modify a memory partition without authorization. This applies to all operations of the different worlds. The authorizations are enforced by the distributed security wrappers and their policies.

The non-secure world components cannot access resources from the secure worlds, however this restriction is not applied to the secure worlds. For example, an application running in a secure world could delegate some computation to a non-secure hardware accelerator. In this case, after the end of processing the IP would be automatically reset by its security wrapper to prevent the misuse of sensitive data. This rule is also applied for cache partitions which are flushed when switching from one world to another. This reduces the overall performance but provides a better level of security and contributes to the **SF.5** security feature.

4.4 Memory protection

One feature of TrustSoC is to protect memory resources from illegitimate accesses and isolate the partitions of the different worlds between themselves. This is enforced by a security wrapper connected to the AXI4 communication bus and then to the memory. Each request coming from the bus is verified by the security wrapper. It verifies that every request complies with the security policies. It compares the identifiers of the transaction, which component is the sender, and in which world the system is currently operating, using the access rights table belonging to the underlying component. It accepts and forwards the transaction when the rights are verified, otherwise it discards the data and raises an error on the bus through the SLVERR signal.

Table 1: Implementation costs for multiple protected and unprotected hardware accelerators from an AMD-Xilinx Zynq-7000 SoC-FPGA (XC7Z010-1CLG400C).

IP	Strategy	LUT	FF	Fmax (MHz)
Sobel filter	Unprotected	2,783	4,355	219
	Protected	2,795	4,359	212
	Overhead %	+0.4	+0.1	-3.2
ASCON-Masked	Unprotected	2,747	2,545	209
	Protected	2,756	2,547	212
	Overhead %	+0.33	+0.1	+1.4
AES-128	Unprotected	3,048	2,031	128
	Protected	3,057	2,034	126
	Overhead %	+0.3	+0.2	-1.6
Karatsuba-128	Unprotected	2,921	3,061	239
	Protected	2,934	3,063	240
	Overhead %	+0.5	+0.1	+0.5
Montgomery-128	Unprotected	4,903	1,625	102
	Protected	4,915	1,627	101
	Overhead %	+0.2	+0.12	-1.0

4.5 Prototyping and testing

In this subsection, we provide implementation results from an AMD-Xilinx Zynq-7000 SoC-FPGA (XC7Z010-1CLG400C). We used the Xilinx Vivado 2020.2 toolchain to implement the TrustSoC prototype. The distributed security wrappers presented in subsection 4.2 were described in VHDL. We used these distributed wrappers to protect five different IPs from cryptographic and signal processing applications. The implementation results are shown in Table 1. The size of these IPs ranges from 2,747 to 4,903 LUTs. All the hardware IPs we used are found in online repositories. We evaluated our hardware implementations with and without the security wrapper, using as metric the hardware utilization in LUTs, FFs, and the maximum frequency attainable by the design.

As shown in Table 1, the resources overhead induced by the security wrapper in number of LUTs is 0.34% at most and in number of registers, 0.13% at most compared to the baseline implementation costs of the IPs. This resource overhead can be explained by the fact that we add logic to each IP in order to implement our distributed security wrappers. In our experimentation we were limited by the size of the fabric in the AMD-Xilinx Zynq-7000. For example the largest multiplier instances we could fit in this board used operands of 128-bits. However, for cryptography applications it is expected to use up to 512-bits operands. Such larger instances would evidently dwarf the hardware costs of the security wrapper in comparison. The overhead in terms of maximum frequencies of operation is not significant. Indeed, the security wrapper does not affect the critical path of the hardware accelerators and thus it does not affect the maximum attainable frequencies. We suspect that the fluctuations that appear in Table 1 are due to the non-deterministic nature of the synthesis process. In conclusion, the security wrapper has a negligible resources and

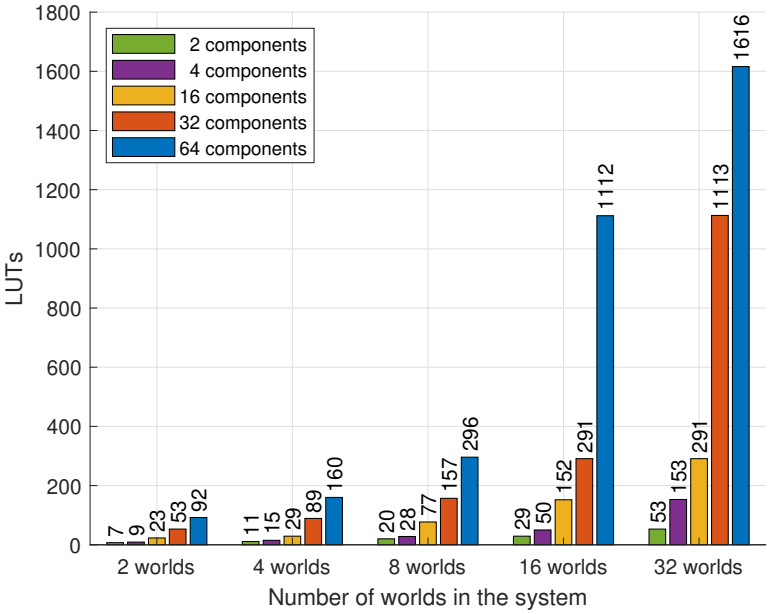


Figure 4: Results of implementation of the security wrapper attached to a BRAM

performance overhead with a significant security improvement.

We then implemented a security wrapper with a BRAM in order to demonstrate the operation of our system and the costs of the world partitioning on a memory block. We implemented this BRAM security wrapper and tested it with a varying number of worlds (2,4,8,16,32) as well as a varying number of IPs which requested access to the memory (2,4,16,63). Fig. 4 shows the results of our implementations. This prototype allowed us to explore the costs and scalability of our proposal.

The overhead in resources is attributed to the size of the access-rights table. This explains the higher overhead for the larger number of worlds with the most IDs. Currently our implementation employs LUTRAMS, but it is also possible to use BRAMS which would reduce drastically the overhead in the number of LUTs. The results of the timing criteria are not displayed since the variation is negligible. From the results in Fig.4 we can conclude that the costs of the security wrapper and world partitioning impose a small overhead on the protected system, which is very much acceptable with the degree of protection provided by the proposed solution.

5 Discussion and Benchmark

In this section, we review the state of the art on secure-by-design architectures for heterogeneous SoCs. We focus on the works which are the most relevant to our proposal. A qualitative comparison between these references is shown in Table 2.

Table 2: Comparison of SoC architectures which provide protections for cache memory, protections for the communication bus, protections for the programmable resources, segregation into multiple secure domains and make their code available as well as the processor type they utilize. ○ stands for no support, ◐ for support but with limitations and ● is for full support.

Architecture	Type of processors	Programmable resources (FPGA)	Protections for the cache memory	Protections for the bus	Segregation in multiple secure domains
CURE [Bah+21]	RISC-V	○	●	●	3 types of enclaves
Hector-V [Nas+21]	RISC-V	○	●	●	0
ARM TrustZone [ARM]	ARM	◐	●	◐	2 worlds
Embedded policing [HSS18]	ARM	●	○	●	2 from ARM TZ
Our proposition	ARM	●	●	●	N worlds (is only limited by available resources)

The most prominent strategy for protecting SoC-FPGAs is ARM's TrustZone [ARM]. This is a security solution available in AMD-Xilinx lines of SoC-FPGAs. Arm TrustZone divides the resources of the processing system in two different worlds: secure and non secure. This partitioning is then extended to the rest of the SoC: peripherals and memories. And in the case of AMD-Xilinx SoC-FPGAs also to the reconfigurable fabric. TrustZone enforces the policies in the system with an identifier called "NS bit" and some controllers. The security identifier is directly implemented in the AMBA buses. Despite its popularity, TrustZone has shown many vulnerabilities that can be exploited to perform attacks and corrupt the security partitioning [BBA19; BB18; Gro+20]. Moreover, TrustZone being a proprietary solution, is not open source which makes it difficult to improve its implementation.

In [Bah+21], Bahmani *et al.* presented an architecture where there are three different types of software enclaves at different levels in the SoC (user-space, kernel-space and sub-space). They modified a RISC-V processor to support an enclave identifier. The rest of the system was also modified to support the enclave identifier. This identifier was then used in the rest of the system and filtering blocks were put in place to determine legitimate accesses. A hypervisor (secure monitor) was used to configure the permissions. Cure also embedded protections against cache side-channel attacks. Policies were set in place to supervise the cache partitioning, allocation and cache eviction. The proposition was a software security solution for the most part, as there were no programmable resources embedded in the SoC. Thus, we consider their solution as not suitable for SoC-FPGAs since a secure architecture must consider the whole SoC and have both software and hardware countermeasures.

Nasahl *et al.* presented in [Nas+21] a secure architecture called Hector-V. The security of their proposal is based on the distinction of the processors. They use a dedicated processor for the normal applications and a processor dedicated for the secure ones. To differentiate between illegitimate and legitimate communications, Hector-V uses identifiers (core ID, process ID and peripheral ID) and filtering blocks called "wrappers". The processors are modified to embed directly the identifiers. Hector-V uses AXI4 as communication protocol so the identifiers are propagated using the AXI4 user signals. There is also distinction in the SoC communication buses. The design includes two communication systems: one for the data and one for the configuration. In Hector-V, the peripherals are bound to an entity and can only accept request from it. The configuration channel is used to define this entity for each peripheral. A secure monitor is responsible for the configuration and to oversee the operation of the communication between all peripherals and the processors. The authors argue that the duplication of the resources in Hector-V can mitigate cache and micro-architectural side-channel attacks. However, this solution would not be entirely viable on a real use-case where the constraints on resource utilization are high. Also, their proposal does not allow to embed programmable resources which is an essential part of SoC-FPGAs. Furthermore, Hector-V only provides a dedicated processor for secure applications, it does not provide segregation in multiple secure domains.

Hagan *et al.* [HSS18] propose a hardware-based pro-active policing and policy architecture. They deploy hardware-based modules called "security policy engines" at the system communication level. These modules are acting as hardware-coded firewalls with a list of permissions that actively monitor the

AXI4 SoC communication bus. For every incoming request, the security policy engine uses the read and write address channels to determine the legitimacy of the transaction. They can either grant or deny access to the device according to the policies stored in a table. The policies are configured via SELinux and can be updated over time. The system also allows to integrate programmable logic (FPGA). Furthermore, it uses ARM TrustZone to provide the designer with the possibility of having a secure domain in their architecture. However, the architecture does not embed security solutions for the cache memory and the user is restricted to two worlds with a single secure one.

6 Conclusion

This paper's main contribution has been to present a novel way of segregating SoC-FPGA into different secure domains based on the ARM TrustZone technology. The security is then enforced with operating policies that prevent any unwanted behavior. To enforce the policies, each hardware resource is given an hardware-coded identifier that cannot be changed. The hardware resources are also given a table of permissions that specify the hardware resources that have the privileges to access the resource. The permissions list is hardware-coded and configured at boot by a software secure configuration. After the initial configuration the permissions cannot be changed. Small distributed communication controllers are then used to monitor the SoC communication bus and determine, according to the resource's policies, the legitimacy of every transaction.

The proposed architecture has been prototyped on an AMD-Xilinx Zynq-7000 SoC. Our experimentation demonstrated that the hardware overhead of the communication monitoring are small in relation to the sized of the purported application domains. The operational frequency of the system would not be affected, and only a small latency overhead is incurred. In regards to memory protections, we have shown that there is a lineal relationship between the hardware overhead and the number of secure worlds and components considered in the system. This can be mitigated with the use of dedicated memories available in most modern platforms.

This paper has laid the foundation for a trusted heterogeneous SoC architecture secure-by-design called TrustSoC. We have demonstrated that security cannot be an add-on functionality and must be carefully thought out from the moment of conception of the architecture. This also must be a dual security model: hardware and software. As our experimental evaluation of the hardware overhead has shown, with a hardware prototype of TrustSoC, this work conduces to the proposition of an efficient and light secure-by-design architecture. It is also a scalable architecture defined at the design stage by the designer.

Acknowledgements

This work is conducted in the framework of the TrustSoC project funded by the French *Agence de l'Innovation et de la Défence* (AID). The authors acknowledge the support of the French *Agence Nationale de la Recherche (ANR)*, under grant ANR-19-CE39-0008 (project ARCHI-SEC)

References

[AF04] Tiago Alves and Don Felton. "Trustzone: Integrated hardware and software security". In: *Information Quarterly* 3.4 (2004), pp. 18–24.

[AMD14] AMD-Xilinx. *Programming ARM TrustZone Architecture on the Xilinx Zynq-7000 All Programmable SoC*. User Guide UG1019 (v1.0). Santa Clara CA, USA: AMD-Xilinx, 2014.

[ARM] ARM. *TrustZone for Cortex-A*. Arm — The Architecture for the Digital World. URL: https://www.arm.com/technologies/trustzone-for-cortex-a (visited on 07/08/2023).

[ARM03] ARM. *AMBA AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite*. 2003.

[Bah+21] Raad Bahmani et al. "CURE: A Security Architecture with CUstomizable and Resilient Enclaves". In: *Proceedings of the 30th USENIX Security Symposium*. Berkeley CA, USA, Aug. 2021. ISBN: 978-1-939133-24-3.

[BB18] El Mehdi Benhani and Lilian Bossuet. "DVFS as a Security Failure of TrustZone-enabled Heterogeneous SoC". In: *IEEE International Conference on Electronics, Circuits and Systems*. Dec. 2018. DOI: 10.1109/ICECS.2018.8618038.

[BB21] Lilian Bossuet and El Mehdi Benhani. "Security Assessment of Heterogeneous SoC-FPGA: On the Practicality of Cache Timing Attacks". In: *Proceedings of the 19th IFIP/IEEE International Conference on Very Large Scale Integration*. IEEE, Oct. 2021, pp. 1–6. DOI: 10.1109/VLSI-SoC53125.2021.9607012.

[BBA19] E. M. Benhani, L. Bossuet and A. Aubert. "The Security of ARM TrustZone in a FPGA-Based SoC". In: *IEEE Transactions on Computers* 68.8 (Feb. 2019), pp. 1238–1248. DOI: 10.1109/TC.2019.2900235.

[Fit] Asa Fitch. *Arm-Based Chips Make Inroads With Apple, Amazon*. British chip-design company puts pressure on Intel while preparing for IPO. The Wall Street Journal. URL: https://www.wsj.com/articles/arm-based-chips-make-inroads-with-apple-amazon-11674436002 (visited on 01/23/2023).

[Gro+20] M. Gross, N. Jacob, A. Zankl and G. Sigl. "Breaking TrustZone memory isolation and secure boot through malicious hardware on a modern FPGA-SoC". In: *Journal of Cryptographic Engineering* (2020). DOI: 10.1007/s13389-021-00273-8.

[HSS18] Matthew Hagan, Fahad Siddiqui and Sakir Sezer. "Policy-Based Security Modelling and Enforcement Approach for Emerging Embedded Architectures". In: *IEEE International System-on-Chip Conference*. 2018. DOI: 10.1109/SOCC.2018.8618544.

[Nas+21] Pascal Nasahl, Robert Schilling, Mario Werner and Stefan Mangard. "HECTOR-V: A Heterogeneous CPU Architecture for a Secure RISC-V Execution Environment". In: *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, May 2021, pp. 187–199. ISBN: 9781450382878.

[OST05] Dag Arne Osvik, Adi Shamir and Eran Tromer. "Cache Attacks and Countermeasures: The Case of AES". In: *Proceedings of the Cryptographers' Track at the 2006 RSA Conference*. Springer, Feb. 2005, pp. 1–20.