On the Security Evaluation of the ARM TrustZone Extension in a Heterogeneous SoC

El Mehdi Benhani[(1)], Cédric Marchand[(2)], Alain Aubert[(2)], Lilian Bossuet[(2)]
Hubert Curien Laboratoty
University of Lyon, Saint-Etienne, France
Emails: [(1)] elmehdi.benhani@univ-st-etienne.fr,
[(2)] firstname.lastname@univ-st-etienne.fr

***Abstract*—As the complexity of System-on-Chip (SoC) and the reuse of third party IP continues to grow, the security of a heterogeneous SoC has become a critical issue. In order to increase the software security of such SoC, the TrustZone technology has been proposed by ARM to enforce software security. Nevertheless, many SoC embed non-trusted third party Intellectual Property (IP) trying to take the benefits of this technology. In such case, is the security guaranteed by the ARM TrustZone technology reduced by the heterogeneity of SoC? In order to answer to this question, this paper presents relevant attack scenarios based on third party IP to exploit some security failures of the TrustZone extension through the all SoC. At the end, this article proposes to SoC designers to consider some design solutions to limit the impact of a malicious IP.**

***Index Terms*—ARM TrustZone, Embedded system security, AXI bus, Hardware Trojan.**

I. Introduction

Many embedded system from mobile systems and automobiles to industrial systems and home appliances are now connected to at least one type of network or service. They store and exchange a large amount of user data, including the sensitive information such as mobile banking details or private information. This cloud of data with the digitization of our world make the internet of things (IoT) an attractive field for attackers, which raises the need for security.

Today, the security of System-On-Chips (SoCs) for IoT is not a straightforward subject. Indeed SoCs are becoming more and more complex, they include many sophisticated applications such as hardware acceleration. Making those applications and the entire SoC secure is a great challenge for semiconductor industry, especially if their designers want to reduce area cost, and power consumption.

The semiconductor industry offers some security solution and one of them ARM TrustZone [1] technology. It is an efficient solution for creating a root of trust and meeting market expectations. The TrustZone is a hardware security extension built over the processing system. It partitions both hardware and software resources into two worlds, one secure where high-value code and data can be protected and one non-secure. The combination of this technology, secure boot and trusted software makes it possible to create a Trusted Execution Environment (TEE) providing isolated execution and integrity of trusted applications.

Several trusted systems for embedded devices follow GlobalPlatforms API standard [2] [3] to enable their TEE to deliver a common security capability across platforms, such as Sierraware's SierraTEE [4], OP-TEE [5] and TOPPERS Project's SafeG [6].

In this paper we present a security evaluation of the TrustZone technology using Xilinx Zynq-7010 FPGA SoC, a wide known SoC platform by research community. The Zynq-7010 SoC is a relevant platform to prototype the propagation of the TrustZone outside the processor. It is TrustZone-enabled SoC with the capability of propagation of the TrustZone concept into its FPGA. Using Xilinx Vivado, designer can partition the FPGA design into secure world and non-secure by affecting a static security status to each Intellectual Property (IP) in the design.

This paper presents some hardware attacks that exploit vulnerability created by the propagation of the TrustZone into FPGA in a heterogeneous SoC. It presents also some countermeasures and design recommendations to avoid a number of security failure.

In the rest of the paper, we first present the state of the art in section 1. The Xilinx Zynq-7010 FPGA SoC [7] is described in section 2, followed by the security evaluation in section 3. Section 4 represents some countermeasure and design recommendations and the conclusion is in section 5.

II. State of the art

The TrustZone has been used to ensure security for diverse use cases including authentication, payment, content protection and companies. In academic research, it is a hot topic that is receiving more and more attention.

Zhang N. et al. [8] presented a study on cache timing-based information leakage of ARM TrustZone. Using the Prime and Probe cache attack, they succeed to recover the full key of an AES software implementation based on T-table. In this attack, the attacker fills the cache with known states in normal world before the execution of the cryptographic operation in secure world and observes the changes in these cache states. They explained that the leakage is due to a fundamental design choice of the TrustZone-enabled cache architecture, which aims to improve system performance by allowing two worlds to share the same cache hardware.

978-1-5386-4034-0/17/$31.00 ©2017 IEEE

Jyothi V. et al. [9] proposed a novel methodology of FPGA TrustZone (FTZ) to incorporate security into the design flow in order to detect and isolate malicious hardware Trojan inside the FPGA. FTZ uses trojan detection based ring-oscillator, and Xilinx Isolation Design Flow (IDF) methodology [10] to identify and isolate anomalies.

Carru P. [11] showed that the Rowhammer effect from a non-secure world can be used to attack a TrustZone secure world. He succeeded to recover a private keys stored in secure memory of an RSA signature implementation, by making a specific access read of a row in the dynamic random-access memory (DRAM). With the bit flip caused by the Rowhammer effect, he bypassed TrustZone protection which prevents non secure software from writing to secure memory.

Roseberg D. [12] studied a vulnerability affecting qualcomms implementation of the TEE (QSEE). The vulnerability was present on a wide variety of android mobile devices supporting TrustZone and utilizing Qualcomm Snapdragon SoC. He succeeded to execute non-secure code in a secure world by using a failure of handling integer overflows in Security Monitor Call (SMC) request function.

The rest of the paper presents our contributions that highlights the propagation of the TrustZone outside the ARM processor in a heterogeneous SoC. In the next section we present our experimental platform.

III. XILINX ZYNQ-7010 FPGA SOC

The Zynq-7010 FPGA SoC is partitioned into Processing System (PS), and Programmable Logic (PL). The PS integrates a dual Cortex-A9, an On-Chip Memory (OCM), a central interconnect, a programmable logic to memory interconnect, a timer controller, a DDR3 controller, and an I/O peripherals. The PS includes also the TrustZone hardware set used to protect the memories and peripherals.

The cortex-A9 in the PS has been used in many projects to run trusted application running over a TEE framework, such as OP-TEE [5] and TOPPERS Project's SafeG [6]. The cortex-A9 provides two virtual cores, one secure and one non-secure accompanied with a monitor mode which is a gatekeeper between the two worlds. The security status in the SoC is related to the Non-Secure (NS) bit in the Secure Configuration Register (SCR) of the Cortex-A9. The PS proposes also a set of configuration registers that can be used from software to control the TrustZone hardware and create a secure custom design, they can be dynamically programmed during runtime execution.

For more details about software and hardware implementation of the TrustZone on the Zynq-7010, the TOPPERS Project [6], Xilinx documentation [13] [14], our implementation tutorial [15] and ARM TrustZone software example [16] provide valuable information and give some recommendations helping to make the system more secure.

The communication between the PS and the PL is ensured by Advanced eXtnsible Interface (AXI) bus [17] which is part of the ARM Advanced Microcontroller Bus Architecture (AMBA) specifications. In the Zynq-7010 SoC, the AXI bus establishes the communication between a master and a slave interface using five separate channels as presented in Fig. 1: Read Address, Read Data, Write Address, Write Data, and Write Response.

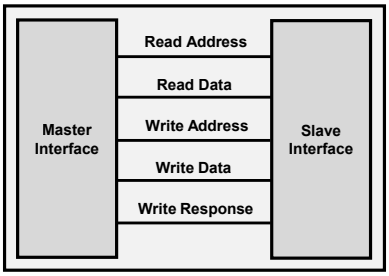

Fig. 1. Communication channels between master and slave interfaces.

Each channel uses a VALID and READY handshake signal pair to indicate when the receiver is ready to process bus data, and to mark when valid data is on the bus. To transmit any signal (address/data/response), the VALID of the sender and the READY of the receiver should be active.

The read and write address channels on the AXI bus have a 3-bit security signal, called AWPROT[2:0] for the write, and called ARPROT[2:0] for the read. The bit number 1 of those protection signals is known as the Non-Secure (NS) bit, its security status is related to the Cortex-A9 NS bit.

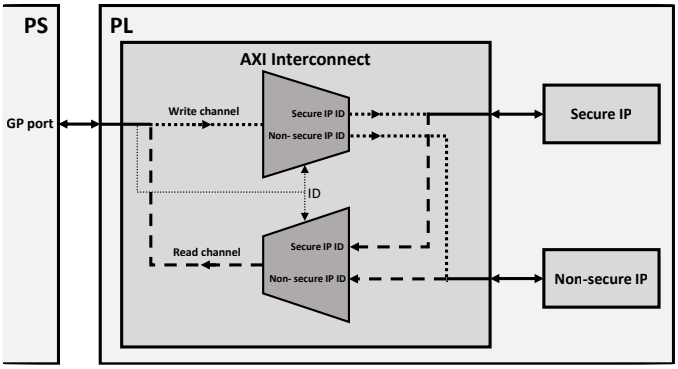

Fig. 2. AXI Interconnect in the PL interfaces.

The AXI Interconnect [18] is a multiplexer-based IP (as shown in Fig. 2) connecting one or more AXI memory-mapped master devices to one or more memory-mapped slave devices. Using Vivado, the security feature on the AXI Interconnect can be statically enabled or disabled and each third-party slave IP cores in the PL can be assigned secure or non-secure status. The IP security status is an AXI Interconnect parameters. The NS bit checking feature can be done on the AXI Interconnect or directly on the third party IP.

If a non-secure master attempts to access a secure slave, a SLVERR (slave error) or a DECERR (decode error) is raised by the slave or the bus [1].

The slave response for a read transaction is transmitted using the RRESP signal of the read data channel and for the write transaction is transmitted by the BRESP signal of the respond

channel. If the transaction is successful, the Slave sends back an OKAY. In the case of a failure, the PS will launch an abort exception which will be treated by the software.

The AXI Interconnect and the protection signals of the AXI bus are the main TrustZone hardware set up in the PL. They propagate the technology concept to the PL by dividing the hardware into two worlds and prohibit the non-secure world from using secure resources.

IV. Security evaluation

This section presents the hardware attacks realized on the Zynq-7010 SoC in order to evaluate the propagation of the TrustZone to the PL. We will not give more details about the hardware malicious modification and the way is implemented, we will only concentrate on the principle of its use. We supposed for all the attacks that the attacker has access to the RTL code using Xilinx tools and he can modify it. We suppose also that the AXI Interconnect is connected to general purpose port (GP port) [14] of the PS and the memory AXI Interconnect is connect to the high-performance port (HP port) [14]. Some of the proposed attack can be generalized to other type of platform, but it could demand sophisticated technics like fault injection [19].

A. Attack#1: Using a secure IP from non-secure world

In this first attack, the attacker makes malicious modification in the design in order to make use of an IP declared secure in the AXI Interconnect from a non-secure world. This attack targets an IP with a memory-mapped slave. Depending on its purpose, it can control the read operation from the IP, the write operation or both.

The PS propagates its security status through the AXI Full bus. If the PS is secure, the AWPROT/ARPROT on this bus are LOW, in another case they are HIGH. The AXI Interconnect checks the security status of the AWPROT/ARPROT signal using a simple conditional test and sends back the result to the PS using RRESP/BRESP signals of the AXI full bus.

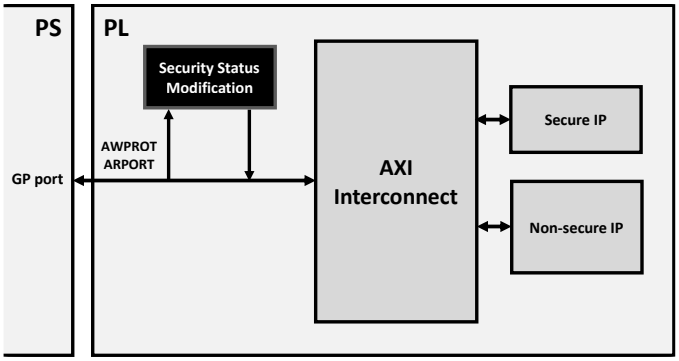

Fig. 3. The addition of a malicious security status modification in the PL.

The secure PS has access to all PL IPs because for the TrustZone technology a software application running in the secure world has access to all design resources. The attacker will make use of this aspect and put its malicious modification before the input of an AXI Interconnect that connects IPs with a slave ports as illustrated Fig. 3. The modification fixes the AWPROT/ARPROT signals to LOW in order to make all the operation coming from the PS secure despite the real security status of the PS. Fixing AWPROT/ARPROT signals targets both operation read and write.

Any operation from the PS non-secure world is secure for the PL and the security status checking operation in the AXI Interconnect does not send back a SLVERR or a DECERR error. The PS works normal without launching an abort exception.

One thing that makes this attack possible is that the PL doesnt share the information about security status of PL IPs with the PS. This information can help securing the slave memory region access using the TrustZone PS hardware.

B. Attack#2: Prohibit the use of a secure IP

In this second attack, the attacker makes a Denial of Service (DoS) by prohibiting access to any secure IP in the design. This attack targets an IP with a memory-mapped slave. The attacker keeps the same location of the malicious modification in attack#1 and the same targeted signals (AWPROT/ARPROT) but for attack#2 scenario, the modification fixes the protection signals to HIGH.

Any operation from the PS is seen as non-secure for the PL. For any attempt to access a secure IP from both PS world, the AXI Interconnect sends back a SLVERR or a DECERR error and an abort exception is triggered in the PS. For the design lifetime, all access to the secure IPs with a slave port connected to an AXI Interconnect with this malware input are condemned.

C. Attack#3: Reply OKAY all the time

In this third attack, the PL responds OKAY to all AXI full bus transaction. The attacker keeps the same location as previous attacks as presented Fig. 4 but it will try to change the response sent by the AXI Interconnect. Depending on the attacker purpose, the modification can affect only the read operation, only the write operation or both.

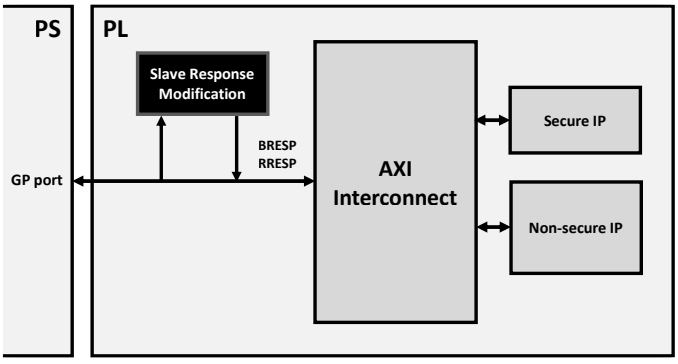

Fig. 4. The addition of a malicious slave response modification in the PL.

The RRESP and BRESP are a 2-bit signals. The attacker tries to make all the transactions seen as successful transaction

by the PS. The attacker fixes the response signals at the value 00 which is interpreted as an OKAY by the AXI Interconnect. The PS is not informed about a faulty access.

If the attacker tries to access a secure IP from non-secure world, the malicious modification covers the error SLVERR or DECERR. The PS doesnt trigger an abort exception and the malicious transaction is stopped at the security checking of the AXI Interconnect.

D. Attack#4: Causing an AXI bus error

In this fourth attack, the PL sends an error respond back to the PS each time the PS makes use of the PL. The malicious modification is made on the same location and targets the same respond signals as for previous attacks. This attack targets only an IP with a memory-mapped slave ports. The attacker chooses to cause one of the two possible errors, the SLVERR error by fixing the BRESP/RRESP signals to the value 10 or the DECERR by fixing the signals to the value 11. Each time the PS tries to use the PL an abort exception is launched. All the IPs with slave port connected to an AXI Interconnect with this malware input cannot be used in the device lifetime.

E. Attack#5: FIFO attack

In this fifth attack, the attacker spies on the secure transaction. The attacker uses a FIFO to store the secure data while the secure world running and recovers it from non-secure world.

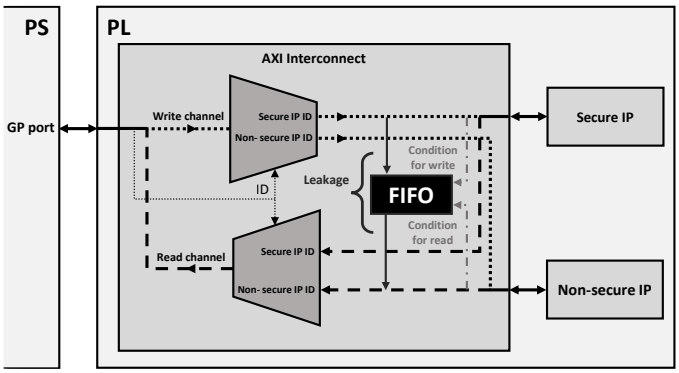

Fig. 5. FIFO malicious insertion on the AXI Interconnect.

The attacker makes malicious modification to the AXI Interconnect as illustrated Fig. 5, and uses a malware non-secure IP. Nicole Fern et al. [20] have already presented a FIFO attack in their works, they give more details about the FIFO integration into the AXI Interconnect and they explain how to choose a read and write conditions for the FIFO. In our case we suppose that the attacker chooses the Ready/Valid signals (handshake signals) in the write channel of the secure bus as FIFO write condition, and the Ready/Valid signals in the read channel of the non-secure bus as read condition.

From the secure world, the system sends secure data targeting the secure IP. In the same time, the FIFO is activated due to handshake in the secure write channel and starts storing data to the FIFO. The size of leaked data depends on the depth of the FIFO and transaction duration. Once the secure transaction finished, the attacker gets the FIFO data by trying a read on the non-secure IP from the non-secure world. The handshake signals in read channel of the non-secure bus activates the read from FIFO.

Another possible scenario of the FIFO attack is to send the secure data after the end of the transaction to be treated by the non-secure IP. For this, the attacker has to use the handshake signals in the write channel of the non-secure bus for read condition, and make a write to the non-secure instead of a read.

In this attack, the attacker can also chose a condition to stop storing data into the FIFO, as example the WLAST signal which indicates the end of a transaction in AXI protocol. The success of the FIFO attack depends enormously on the depth of the FIFO. But if the attacker is spying on a cipher implementation in the PL, a small FIFO to get a part of the key can help to guess a full key.

F. Attack#6: The master port attack

In this sixth attack, the attacker uses a malicious IP with a memory-mapped master port to have a direct access to DDR3 as presented Fig. 6. As an example of a malicious IP, the attacker can use a hardware cipher implementation or a hardware block that treat a huge amount of data. In general, the aim of using an IP with a master port is to accelerate treatment without perturbing the processor. The attack can also be realized using a Direct Access Memory (DMA) without a security configuration from the TEE. The DMA allows the attacker to make copy from secure storage to non-secure.

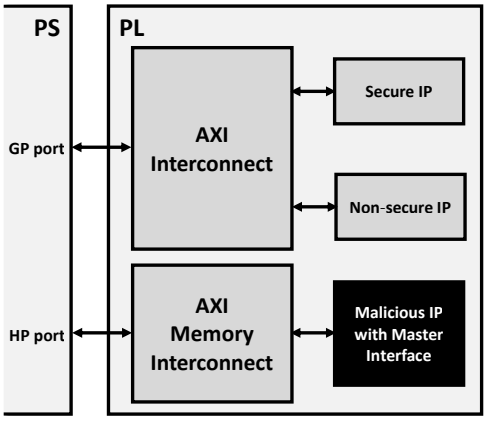

Fig. 6. Malicious IP insertion with a memory mapped master.

If the master interface has a direct access to memory while processor is running, the attack bypasses almost all security measures [21] of the processor and has the ability to read encryption keys from secure world storage, install malware in secure memory, or control other secure IP. A master interface in a design has the capability of controlling the security status of the connected AXI bus.

The success of this attack depends on two elements. The first element is the mapping of the master interface in the

memory, the mapped region should cover the targeted range of memory where the secure data is stored or where to install the malware. The second element is depend on the attacker knowledge about the software mapping and the features of the targeted platform.

For the Zynq-7010, DDR memory is divided into 64 MB section [14] which the developer of the TEE sets secure or not using a configuration register. If the attack starts by reading this register, the attacker will know about the security memory mapping and use it to have an easy access to the secure memory in the device.

For Intel Cyclone V [22], the memory partitioning is easy and more advantageous. It allows to choose some region of the memory and declare it as secure or not by creating some rules. It doesnt upset with an already defined section size.

V. Countermeasure and design recommendations

This section represents some countermeasures and design recommendations to help secure the system against some of the presented attack. But, they dont protect the system from high-level physical attack as shown in [19]. A designer shouldnt make use of the connection automation or routing automation proposed by many design creator tools. The automation tools help to gain time and resource but it does not allow a mastery of the design.

Some of the design recommendations will stop the TrustZone protection feature from propagating to the PL, but they will let a designer to create a secure propagation of the TrustZone concept which is dividing the hardware design into two worlds.

Design recommendation#1: The PS provides a register configuration [14] that can stops all the non-secure transaction from propagating to the PL through the GP port. The designer can use those registers and the two GP ports in the Zynq-7010 in order to propagate safely the TrustZone concept into FPGA.

First, the designer needs to create a design with two AXI Interconnect, one connected to secure IPs and a second to non-secure IPs as illustrated Fig. 7. After while creating the TEE, the designer should set the GP port connected to the secure AXI Interconnect to prohibit non-secure transaction from reaching the PL. The designer should not declare the PL IPs as secure on the secure AXI Interconnect because it can be used as attack vector and exploited using the attack#2.

If the designer follows the previous steps the security status checking on the system will be done on the PS part. The checks doesnt use only the AXI Interconnect verification, but others PS TrustZone hardware that protect memory access. This Design recommendation targets TrustZone-enabled SoC with FPGA such Zynq-7010 and Cyclone V which admits the same TrustZone architecture and implementation way as Zynq-7010.

Design recommendation#2: The PS provides also a register configuration [14] that can set the HP port to accept only the non-secure transaction coming from the PL and targeting the external memory. The Zynq-7010 offers four HP ports. The designer can set one of the four as non-secure port and

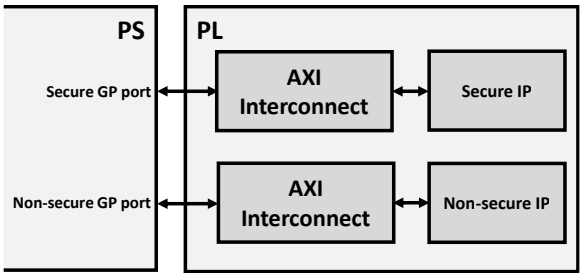

Fig. 7. Creation of two world using two AXI Interconnect.

connect to all non-sensitive memory-mapped master interface. This setting will provide the IPs with direct access to memory from acceding secure area in the external memory. As in the Design recommendation#1, the security checking doesnt use only the AXI Interconnect verification, but others PS TrustZone hardware that protect memory access. This second Design recommendation targets TrustZone-enabled SoC with FPGA such Zynq-7010 and Cyclone V.

This Design recommendation doesnt cover the use of a memory mapped master in order to access a secure memory area, it needs to be enforced using one of the following countermeasures:

Countermeasure#1: The Xilinx memory protection unit (XMPU) [23] which is available only on the Zynq Ultrascale+. It provides memory protection, it can be dynamically configured to isolate a master or a given set of masters to a programmable set of address ranges.

Countermeasure#2: Secbus [24] which uses a hardware cipher block between the AXI Interconnect and the memory controller. It applies cryptography operation to all read and write to the external memory. The team project developed a software library to use [25] the cipher hardware.

Countermeasure#3: Input-Output Memory Management Unit (IOMMU) [26] which is a memory management unit (MMU) that connects a Direct-Memory-Access-capable (DMA-capable) I/O bus to the main memory. It works like the MMU. The IOMMU maps device-visible virtual addresses to physical addresses. It provides also memory protection from faulty or malicious devices. It can be programmed from the TEE.

Another important security protection is the secure boot [27] that many platform provides. We need to use it in order to program all the sensitive registers during the first boot loader, for example configure the register for our countermeasure.

VI. Conclusion

In this paper we presented a security evaluation for the TrustZone technology propagation to FPGA using the Xilinx Zynq-7010 SoC. We presented six attacks using small malicious modification in order to access secure data, or create a DoS. Some of those presented attacks can be generalized to other type of SoC, but it requires sophisticated technique to realize.

TABLE I
ATTACKS AND THEIR PREVENTIONS

	Design recommendation#1	Design recommendation#2	Countermeasure#1	Countermeasure#2	Countermeasure#3	No prevention
Attack#1	**X**					
Attack#2						**X**
Attack#3						**X**
Attack#4						**X**
Attack#5						**X**
Attack#6		**X**	**X**	**X**	**X**	

We presented also some countermeasures and design recommendations in order to help designers to prevent some of the six attacks and protect the system.

VII. Acknowledgement

This work was carried out in the framework of the FUI-AAP20-Project TEEVA supported by Bpifrance.

References

[1] A. ARM, "Security technology building a secure system using trustzone technology (white paper)," *ARM Limited*, 2009.
[2] GlobalPlatform, "Tee protection profile, globalplatform device committee tee protection profile version 1.2.1," *https://www.globalplatform.org/specificationsdevice.asp*.
[3] ——, "Tee client api specification version 1.0," *http://globalplatform.org*, 2010.
[4] Sierraware, "Open virtualization - arm trustzone and arm hypervisor open source software," *http://www.sierraware.com*, 2017.
[5] OP-TEE, "Project op-tee, github repository," *https://github.com/OP-TEE*, 2017.
[6] S. TOPPERS, *https://www.toppers.jp/en/safeg.html*.
[7] Xilinx, "Zynq-7000 all programmable soc technical reference manual, ug585 v1.11," 2016.
[8] N. Zhang, K. Sun, D. Shands, W. Lou, and Y. T. Hou, "Truspy: Cache side-channel information leakage from the secure world on arm devices." *IACR Cryptology ePrint Archive*, vol. 2016, p. 980, 2016.
[9] V. Jyothi, M. Thoonoli, R. Stern, and R. Karri, "Fpga trust zone: Incorporating trust and reliability into fpga designs," pp. 600–605, 2016.
[10] E. Hallett, "Isolation design flow for xilinx 7 series fpgas or zynq-7000 ap socs (vivado tools)."
[11] P. Carru, "Attack trustzone with rowhammer," *https://firmware-security.com/tag/rowhammer/*, 2017.
[12] D. Rosenberg, "Qsee trustzone kernel integer over flow vulnerability," in *Black Hat conference*, 2014.
[13] Y. Gosain and P. Palanichamy, "Trustzone technology support in zynq-7000 all programmable socs," *Xi1inx, Report*, 2014.
[14] Xilinx, "Programming arm trustzone architecture on the xilinx zynq-7000 all programmable soc user guide, ug1019 v1.0," 2014.
[15] SESAM, "Using trustzone on xilinx zynq soc," *http://perso.univ-st-etienne.fr/bl16388h/tutorial.pdf*, 2017.
[16] A. ARM, "Cortex-a9 trustzone example," *http://infocenter.arm.com/-help/topic/com.arm.doc.faqs/ka15417.html*, 2013.
[17] A. Xilinx, "Reference guide, ug761 v13. 1," *URL http://www. xilinx. com/support/documentation/ip_documentation/ug761_axi_reference_guide.pdf*, 2011.
[18] L. I. P. Guide, "Axi interconnect v2."
[19] N. Timmers, A. Spruyt, and M. Witteman, "Controlling pc on arm using fault injection," in *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2016 Workshop on*. IEEE, 2016, pp. 25–35.
[20] N. Fern, I. San, C. K. Koç, and K.-T. T. Cheng, "Hardware trojans in incompletely specified on-chip bus systems," in *Proceedings of the 2016 Conference on Design, Automation & Test in Europe*. EDA Consortium, 2016, pp. 527–530.
[21] N. Jacob, J. Heyszl, A. Zankl, C. Rolfes, and G. Sigl, "How to break secure boot on fpga socs through malicious hardware."
[22] Intel, "Sdram controller address map, cyclone v hps memory map," 2017.
[23] Xilinx, "System protection unit, chapter in the of the zynq ultrascale+ mpsoc software developer guide, ug1de7 v4.0," 2017.
[24] J. Brunel, R. Pacalet, S. Ouaarab, and G. Duc, "Secbus, a software/hardware architecture for securing external memories," in *Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2014 2nd IEEE International Conference on*. IEEE, 2014, pp. 277–282.
[25] Secbus, "Trescca project," *https://secbus.telecom-paristech.fr/*, 2016.
[26] I. AMD and O. Virtualization, "Technology (iommu) specification," 2007.
[27] L. Sanders, "Secure boot of zynq-7000 all programmable soc," *Application note XAPP1175 (v1. 0), Xilinx*, 2013.