The Vulnerability of ARM TrustZone in an FPGA-Based SOC Environment

Stephen Donchez, *Member, IEEE*

***Abstract*—ARM's TrustZone platform is a well-known platform for the implementation of security functionality in embedded systems. Although the platform is widely used in the larger embedded development industry, it has been less widely studied with regards to its use in Field Programmable Gate Array (FPGA) based embedded systems, which have widespread popularity. This paper analyzes current research work being done in this field, which has demonstrated that there are numerous potential vulnerabilities exposed by such a system. It then analyzes the countermeasures recommended by those conducting said research, as well as proposes avenues for future work.**

***Index Terms*—ARM TrustZone, Embedded System, System-on-a-Chip, FPGA Security**

I. Introduction

IT has rapidly become common knowledge within the technology industry, and to some extent in society at large, that the phrase "secure IoT device" is an oxymoron. Recent advances in computer technology have led to a massive surge of smart devices, with a particularly rapid growth in the consumer electronics sector. However, this rapid proliferation of such devices has led to an unfortunate discovery – many of them fail to adequately address security concerns, leading to vulnerabilities that are often harnessed by malicious actors.

As a result, the industry has begun to take a second look at security in embedded systems. To this end, ARM ltd., the organization that oversees the development of the ARM processor family, has introduced the ARM TrustZone platform, which seeks to offer "an efficient, system-wide approach to security with hardware-enforced isolation built into the CPU." [1] This platform effectively partitions the processor into two discrete "worlds", one for secure operations and one for nonsecure (or normal) operations.

TrustZone has enjoyed tremendous success since its inception, and forms the basis for the security of many common devices, including Android based smartphones. However, the concept of Field Programmable Gate Array (FPGA) based System-on-a-Chip (SoC) devices introduces a host of complexities into the implementation of a TrustZone enabled system. The presence of in-situ reprogrammable hardware in such a system creates drastically increased potential for malicious actors to attempt to compromise the integrity of said system.

Beyond the vulnerability created by having logic that can be altered present in such a system, the FPGA development process introduces several additional concerns. First and foremost, the industry as a rule relies heavily on third party logic designs, known as intellectual property (IP), for abstracting much of the intricacies of these complex systems. The presence of this IP brings with it a host of potential security concerns, both as a result of maliciously compromised IP and also defects in otherwise genuine IP that may expose the larger system to exploitation. Furthermore, the nature of the FPGA development process is heavily automated by a complex buildchain not dissimilar to a compiler. This poses another avenue for attack - compromised buildchain software could result in a device that performs as expected but presents additional avenues for exploitation.

S. Donchez is with Villanova University, Villanova, PA 19085 USA e-mail:sdonchez@villanova.edu

Manuscript received May 6, 2020

A. Structure of the Paper

This paper is structured into 7 major sections. The introduction in Section I seeks to provide essential context as to the purpose of the ARM TrustZone platform, as well as the additional complexities introduced by using the platform in combination with a FPGA based SoC. Section II provides an explanation of the principles governing the operation of the TrustZone platform itself. Section III and IV discusses the state of current research on the vulnerabilities related to the implementation of the ARM TrustZone into both traditional and FPGA-based Embedded Systems, while Section V describes mitigation strategies outlined in this research. Section VI discusses potential avenues for further research. Section VI concludes the paper.

B. A Comment on the State of the Industry

Serious consideration of security in FPGA-based Embedded Systems is a novel field for research. There are surprisingly few scholarly articles regarding this field, from an even smaller pool of researchers. Accordingly, this paper will dar extensively from other works on the TrustZone Architecture, many of which may not directly concern FPGA-based devices.

II. Details of the TrustZone platform

ARM's TrustZone technology exists to facilitate a Trusted Excecution Environment (TEE), in order to enable the execution of "sensitive" tasks in such a way that maintains their integrity without imposing massive constraints on system design. It facilitates this TEE by means of partitioning the Processing System (PS) into two distinct "worlds" - the secure world and the normal (interchangably referred to as non-secure) world. The technology then polices access to secure world resources to ensure that non-secure tasks aren't making erroneous (or malicious) accesses to secure world resources.

This is acomplished by the system's "Monitor Mode", which performs context switches between worlds while also verifying access permissions and the validity of secure-world directed requests.

A. Securing the AXI Bus

TrustZone extends the concept of worlds beyond the PS and into peripherals, via an extension to the AMBA3 Advanced eXtensible Interface (AXI) bus protocol. This ensures that non-secure bus masters don't have access to secure slaves, while enabling secure masters to access both secure and non-secure slave devices. Furthermore, this is accomplished without requiring any modifications to the peripherals themselves, enabling compatability with pre-existing devices.

The principle means of enforcing isolation on the AXI (system) bus is through a pair of bits contained in the extended bus protocol. These bits are termed the Non-Secure (or NS) bits, and one is dedicated to indicating the status of read operations, while the other is reserved for write operations. [2] In the case of a non-secure bus master, these bits are permanently hardwired high, thereby preventing access to any secure world resources. Meanwhile, secure world bus masters drive these lines using internal logic, allowing them access to both secure and nonsecure resources.

Attempts to access secure world resources by a non-secure master result in a failure mode that is implementation dependent. The sytem can either allow the request to fail silently or can raise one of two errors: a slave error or a decode error. These errors are raised on the BRESP (write) and RRESP (read) response signals, another crucial part of the TrustZone's AXI controls. [3]

B. Securing the Advanced Peripheral Bus

The AXI bus serves to connect the ARM processor with a variety of other components internal to the system, much like the PCI family of busses commonly found in laptops and desktops. However, most peripheral devices do not require the high rates of data transfer (and the correspondingly stringent constraints) of the AXI specification. Instead, these peripherals reside on the Advanced Peripheral Bus (APB), and are interfaced with the rest of the system via the AXI-APB bridge. [4]

As peripherals are often outside the scope of the designer's control, backwards compatability is an impoerant factor in system design. Therefore, the APB specifications do not allow for the implementation of an equivalent mechanism to the NS bit, as any such implementation would not be compatible with prior versions of the AMBA protocol. To mitigate this, the TrustZone architecture allocates responsibility for security enforcement of APB peripherals to the AXI-APB bridge itself.

The bridge features a set of one-bit input signals, one per peripheral, used to indicate their respective security states. These signals can either be tied statically to limit the peripheral to one world or the other, or can be driven by some other logic to allow for dynamic reassignment. One example of such logic is ARM's TrustZone Protection Controller (TZPC) [4], although developers are also free to develop their own control system.

C. Securing the AXI Switch

The nature of the AXI protocol is such that any non-trivial set of bus devices will require the implementation of a routing mechanism to facilitate traffic flow. For Xilinx based systems, this mechanism is the AXI Interconnect, a Xilinx provided piece of IP. [5] The interconnect provides a crossbar switch to enable routing between multiple masters, as well as a host of converters to handle different implementations of the protocol (data widths, clock speed, protocol versions, etc.). Altera offers a similar piece of IP, the Platform Designer (formerly Qsys) Interconnect, which offers identical capabilities as well as an interface with their Avalon streaming protocol. [6]

The TrustZone documentation does not outline specific treatment of these interconnect devices, however research explored in sections III and IV below demonstrate that substantial vulnerabilities exist in this logic. As all bus traffic in the system must flow through this IP, it is imperative that it be adequately secured.

D. Securing the System Memory

While all of the above aspects of the system concern the security of either peripheral devices or other components likely located on the programmable logic (PL) side of the system, the main memory is another aspect of the system that requires careful consideration. Although the main memory is accessed via the AXI bus the same way any other system component is reached, it is unique in that it contains both secure and non-secure elements.

Like the AXI Interconnect, the implementation of the Memory Management Unit (MMU) is manufacturer specific. However, both Intel/Altera and Xilinx's offerings provide compatability with the TrustZone concept of NS bits, and verify security status as well as the observance of traditional process isolation. [7]

Additionally, the memory is heavily integrated into the caching system present in almost all modern devices. As the fundamental premise of caching is speed of access, the tradeoff between security and ease of access is highly present in this aspect of the system's design.

III. Related Research in Embedded Systems

The TrustZone Architecture has seen an explosion in popularity over the past several years, prompted partially by the maturation of the industry but also by ARM's release of much of the previously protected details of the system's implementation into the public domain. Accordingly, a diverse body of research surrounding the system has been authored. This research has revealed a number of vulnerabilities inherent in the platform, regardless of if it is implemented on a FPGA-based or traditional system.

A. Architectural Issues

B. Implementational Issues

C. Hardware Issues

IV. Related Research in FPGA-Based Embedded Systems

In addition to the various vulnerabilities inherent in the TrustZone architecture outlined above, the presence of a PL in a FPGA-based SoC introduces a number of additional concerns. As mentioned earlier, critically small amounts of research have been conducted in this field, however several papers have been presented that discuss these vulnerabilities in depth. These papers focus primarily on specific vulnerabilities present in SoC systems, as opposed to the general discussion of the platform often found in references in the preceding section. Several such discussions are outlined below.

A. Attacks on the AXI Bus NS Bits

The concept of the NS bits (Formally the AWPROT and ARPROT signals) as they relate to the AXI bus's overall security makes them an attractive target for compromise by actors seeking to violate the integrity of the Secure World. As Benhani et al. demonstrate in [2], two potentia objectives can be achieved through malicious actions on these bits. By altering these bits such that they are no longer tied high (which otherwise prevents access to the Secure World), it is possible for a non-secure component to access Secure World Resources. Alternatively, by taking a previously variable set of NS bits (such as those present on a Secure World component) and tying them high, it is possible to implement a denial of service attack against that resource, impeding the ability of the Secure World routines that utilize that component to do so.

At first glance, a modification such as this seems infeasible, both due to the nature of the access required to implement them as well as the high likelihood of discovery that a suddenly nonfunctioning system component would imply. However, this is not necessarily the case. Such vulnerabilities could be exploited easily by either a "rogue designer" or, alternatively, by a compromised element of the buildchain. Furthermore, the nature of the modification is not necessarily such that a device consistenyl behaves in a manner that is indicative of the faulty configuration. Rather, the alteration of these signals could be acomplished by more complex logic, such that the component being attacked behaves as originally intended except under a precise set of circumstances.

This type of vulnerability has always been present in any kind of embedded system via a compromised manufacturing process or inadequate supply chain security. Any of these scenarios could allow for the introduction of a hardware trojan, such as is outlined in [8]. However, the presence of logic that can be reprogrammed, in many cases from within the system itself, makes the potential for the manipulation of these signals much greater.

B. Attacks on the AXI Bus BRESP and RRESP Signals

As an alternative to the modification of the NS bits, Benhani et al. also present an attack vector that focuses on compromising the response signals BRESP and RRESP in [2]. As one might recall, these signals are used to indicate to the requesting device (the Bus Master) if communication between the two devices is permitted based on the NS Bit parameters and the nature of the slave device. Therefore, it is conceivable that these response signals could be altered instead of the inputs to the verification process.

The risk of data leakage posed by the return of a false positive on these signals is somewhat less prominent than in the case of modification of the NS bits, as it would require the master to make an erroneous request that would normally result in an error. However, the risk of a denial of service attack remains present. Additionally, it is worth mentioning that the poossibility of the malicious use of other logic to trigger such an exploit under specific circumstances is equally as valid here as it is in the NS attacks.

C. Attacks on the AXI Interconnect

The AXI Interconnect (or Qsys interconnect for Intel/Altera FPGAs) is another prime target for exploitation, as all bus traffic across the entire system must pass through the crossbar as it flows from component to component. As all the signals (including the NS bits and the response signals) must also flow through the crossbar, this represents another potential means of modifying these bits to achieve the effects described above.

However, there is another, potentially more concerning avenue for compromise in the interconnect logic. As opposed to a Denial of Service style attack, it is possible for a malicious actor to implement a Man-in-the-Middle (MitM) attack by inserting a First In First Out (FIFO) Queue into the crossbar, as is described in [3] and [2]. Such data could then be conceivably accessed by a malicious non-secure IP core, or could even be extracted using a side channel attack such as those suggested in the literature [9].

D. Direct Memory Access Attacks

In [7], Gross et al. outline a number of potential attack vectors which focus on compromising secure elements of the system's main memory by means of compromised hardware contained within the PL. As outlined in Section II,most of these details are manufacturer specific, and [7] focuses heavily on Xilinx systems.

One particularly concerning vulnerability outlined in [7] is the potential to enable the execution of malicious binaries in the Secure World PS by compromising the mechanisms by which the Secure World verifies the integrity of its executables. By manipulating PL components (As outlined in the preceding subsections), the researchers were able to isolate and extract the routine that provided this verification capability, and succeeded in modifying it such that it would always indicate that a routine had been verified.They then were able to redeploy it to the system, and proceeded to demonstrate the execution of their own untrusted binary in the Secure World.

The literature [2] outlines a similar memory vulnerability utilizing the ARM Accelerator Coherency Port (ACP) in conjunction with the TrustZone Components for configuring peripherals and memory. These devices, the TZPC outlined in section II.B and the TrustZone Address Space Controller

(TZASC) designate access to specific peripherals or 64 MB regions of memory, respectively. By manipulating the values in the TZASC, a malicious actor can declare a previously secure memory region non-secure, and then access the data contained therein from non-secure IP.

V. Vulnerability Mitigation Strategies

Several of the works which outline the vulnerabilities discussed above also outline proposals for mitigating the risks posed by these vulnerabilities. Such mitigation strategies are discussed in the subsections below.

A. Predefinition of IP permissions

Many of the above attacks rely on the ability to compromise the set of signals used to enforce world isolation on the AXI Bus. This system is particularly susceptible to compromise in FPGA-Based SoCs, owing to the ease with which the logical elements of the system can be dynamically modified. To this end, Gross et al. propose a novel strategy in [7] - the definition of a security policy table for PL Bus Masters. In essence, this mitigation strategy seeks to define in a central location the status of the various bus devices according to the world they are associated with. Based on this listing, they propose a firewall mechanism that can monitor the flow of traffic to ensure that all devices are truthfully reporting their security status, either preventing or reporting unprivileged escalation or denial of service attacks.

B. AXI Isolation

Taking things a step beyond predefinition of permissions, the authors of [3] propose a modification of the fundamental nature of the PS/PL interaction: the isolation of worlds by seperate AXI Interconnect Logic, rather than a unified logic responsible for its own enforcement. By moving all permission verification processing into the PS, and reaching out to the apropriate AXI Interconnect for the world in question, there is no longer any capacity for malicious logic to alter the security signals and gain unwarranted access, nor can a denial of service attack be implemented.

Although this mitigation strategy provides unparalleled security with regards to the AXI bus, it does so at a tremendous performance cost. The PS must now integrate permissions verification into its schedule, which decreases the amount of time in which it can perform the trusted operations it was intended to do. Furthermore, the complete isolation of worlds on seperate AXI Interfaces means that one of the chief advantages of secure IP is no longer available: said IP can no longer access resources in both worlds. In some cases, this will require a duplication of IP in the two worlds. In others, it will require the PS to perform additional work in order to transfer information between worlds so that all operations requiring secure IP can be done solely in the secure world.

C. Post Synthesis Design Verification

Another key vulnerability inherent in FPGA-Based devices is the extensive toolchain required to design, synthesize, place, and program logic on the device. Because even very simple logic designs can spawn incomprehensible amounts of artifacts, it is often impractical to do a thorough review of all of said artifacts. This makes it possible for vulnerabilities to exist in this code, either through pure error or the malicious actions of a rogue designer or a compromised design tool.

To this end the [2] poses the concept of an automated verification system to check files generated by the synthesis proess are in compliance with the best practices of TrustZone implementation. This would in many ways parallel the feedback provided by the synthesis tools themselves, however such tools are blind to the nature of the code being compiled. This necessitates the use of a (not currently existent) external tool.

VI. Avenues for Future Research in TrustZone Enabled Systems

As the implementation of this architecture on FPGA-Based SoCs is a relatively novel concept, numerous opportunities exist for further research to contribute to the body of knowledge currently existent on the topic. Several of the concepts outlined below are mentioned briefly in some of the existing works discussed previously, while others draw from contempory work done in other similar fields that have applicability to the concerns outlined above.

A. Cache Exploitation

A substantial portion of the Direct Memory Access (DMA) vulnerability outlined in the preceding sections is due to the nature of the relationship between the main memory and the cache. As these systems are tightly connected, and as lines of the cache are frequently overwritten, great potential exists for timing issues or other failures to result in the exposure of secure-world data in the cache to non-secure resources. Accordingly, further research on the behavior of the cache and the logic responsible for controlling it would be useful to provide clarity into the security of the cache when it is accessed not only by the PS but also by elements of the PL.

B. PL characterization and verification

The majority of the vulnerabilities discussed herein are centered around the ability of the programmable logic to be dynamically modified, allowing for the insertion of hardware trojans. In many other fields, the concept of Physical Unclonable Functions (PUFs) have gained notoriety as a means by which to characterize and thereby uniquely identify a piece of hardware. The implementation of such a technology, or a parallel, would enable the Secure World to verify not only the integrity of software (as is already done by signatures) but also the hardware logic associated with a bus device. This would entirely prevent malicious actors from modifying the PL in order to manipulate entry into the secure world.

VII. Conclusion

The conclusion goes here.

ACKNOWLEDGMENT

The author would like to thank Dr. Xiaofang Wang, Villanova University, for her support in the authoring of this paper.

REFERENCES

[1] "TrustZone," library Catalog: developer.arm.com. [Online]. Available: https://developer.arm.com/ip-products/security-ip/trustzone

[2] E. M. Benhani, L. Bossuet, and A. Aubert, "The Security of ARM TrustZone in a FPGA-Based SoC," *IEEE Transactions on Computers*, vol. 68, no. 8, pp. 1238–1248, Aug. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8643768/

[3] E. M. Benhani, C. Marchand, A. Aubert, and L. Bossuet, "On the security evaluation of the ARM TrustZone extension in a heterogeneous SoC," in *2017 30th IEEE International System-on-Chip Conference (SOCC)*. Munich: IEEE, Sep. 2017, pp. 108–113. [Online]. Available: http://ieeexplore.ieee.org/document/8226018/

[4] "ARM Security Technology Building a Secure System using TrustZone Technology," p. 108.

[5] "AXI Interconnect v2.1 LogiCORE IP Product Guide," Dec. 2017. [Online]. Available: https://www.xilinx.com/support/documentation/ip_documentation/axi_interconnect/v2_1/pg059-axi-interconnect.pdf

[6] "Intel Quartus Prime Standard Edition User Guide: Platform Designer." [Online]. Available: https://www.intel.com/content/www/us/en/programmable/documentation/jrw1529444674987.html

[7] M. Gross, N. Jacob, A. Zankl, and G. Sigl, "Breaking TrustZone Memory Isolation through Malicious Hardware on a Modern FPGA-SoC," in *Proceedings of the 3rd ACM Workshop on Attacks and Solutions in Hardware Security Workshop*, ser. ASHES'19. London, United Kingdom: Association for Computing Machinery, Nov. 2019, pp. 3–12. [Online]. Available: https://doi.org/10.1145/3338508.3359568

[8] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, "Hardware Trojan Attacks: Threat Analysis and Countermeasures," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1229–1247, Aug. 2014, conference Name: Proceedings of the IEEE.

[9] S. K. Bukasa, R. Lashermes, H. Le Bouder, J.-L. Lanet, and A. Legay, "How TrustZone Could Be Bypassed: Side-Channel Attacks on a Modern System-on-Chip," in *Information Security Theory and Practice*, ser. Lecture Notes in Computer Science, G. P. Hancke and E. Damiani, Eds. Cham: Springer International Publishing, 2018, pp. 93–109.

Stephen Donchez (M'20) was born in Bethlehem, PA, USA in 1998. He anticipates receiving his B.S. in computer engineering from Villanova University, Villanova, PA in 2020.

In the summer of 2018 he was a software engineering intern at Harris Coroporation (now L3Harris Technologies, Inc.). He returned to L3Harris for the summer of 2019 as a systems engineering intern, and anticpates returning to the same position for the summer of 2020 at their facility in Clifton, NJ. His research interests include FPGAs, embedded software development, and embedded systems with a focus on System-on-a-Chip technologies.

Mr. Donchez is a student member of the IEEE.