Is RISC-V ready for Space? A Security Perspective

Luca Cassano[1], Stefano Di Mascio[2], Alessandro Palumbo[3], Alessandra Menicucci[2],
Gianluca Furano[4], Giuseppe Bianchi[3], Marco Ottavi[3,5]
[1]*Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano*, Italy
[2]*Department of Space Engineering, Delft University of Technology*, The Netherlands
[3]*Dipartimento di Elettronica, Ingegneria dell'Informazione, Università degli Studi di Roma Tor Vergata*, Italy
[4]*European Space Agency*, The Netherlands,
[5]*Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente*, The Netherlands
[1]luca.cassano@polimi.it, [2]s.dimascio@tudelft.nl, [2]a.menicucci@tudelft.nl,
[3]name.surname@uniroma2.it, [4]gianluca.furano@esa.int, [5]m.ottavi@utwente.nl

2022 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT) | 978-1-6654-5938-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/DFT56152.2022.9962352

***Abstract*—Integrated circuits employed in space applications generally have very low-volume production and high performance requirements. Therefore, the adoption of Commercial-Off-The-Shelf (COTS) components and Third Party Intellectual Property cores (3PIPs) is of extreme interest to make system design, implementation and deployment cost-effective and viable w.r.t. performance. On the other hand, this design paradigm exposes the system to a number of security threats both at design-time and at runtime. In this paper, we discuss the security issues related to space applications mainly focusing on threats that come from the adoption of the well-known RISC-V microprocessor. We highlight how Hardware Trojan horses (HTHs) and Microarchitectural Side-Channel Attacks (MSCAs) may compromise the overall system operation by either altering its nominal behavior or by stealing secret information. We discuss the security extensions provided by the RISC-V architecture as well as their limitations. The paper is concluded by an overview of the issues that are still open regarding the security of such microprocessor in the space domain.**

***Index Terms*—Microarchitectural Side-Channel Attacks, Microprocessors, Hardware Security, Hardware Trojan Horses, RISC-V, Space Applications.**

I. Introduction

The idea of employing Commercial-Off-The-Shelf (COTS) components, Third Party Intellectual Property cores (3PIPs) and commercial Computer Aided Design (CAD) tools in space systems is becoming increasingly popular [1], [2]. Indeed, this design choice represents an interesting trade-off between performance and cost for application scenarios where the integrated circuits production volume is extremely low, as the case of space applications. The basic idea is to reuse legacy (i.e., not specifically designed for space) subsystems, components and 3PIPs in space systems to benefit from both reduced development cost and high performance [3].

When specifically looking at microprocessors, designers can either choose to define their own instruction set architecture (ISA) or use an already available ISA. While the employment of open-source software can be considered as a legacy procedure, hardware has not yet fully experienced the disruptive effects of openness. Nevertheless, over the last years the RISC-V architecture has risen in popularity, drawing the attention of several universities and companies previously focusing on other open and free ISAs, proprietary ISAs or even on ISAs designed in-house (with the big drawback of having to design and maintain a software ecosystem) [4].

978-1-6654-5938-9/22/$31.00 ©2022 IEEE

While reducing costs, this design paradigm exposes the obtained system to a number of security threats both at design-time and at runtime. The production of commercial integrated circuits (ICs) is characterized by a globalized supply chain [5]. The benefit of such a globalized supply chain is a reduction of design cost and time that, on the other hand, comes at the cost of a loss of trust in the final ICs [6]. The consequence of is that it is very hard to ensure/assess the trustworthiness of all the parties involved in the supply chain. Therefore, the product is exposed to a number of threats: ICs may be overproduced by the foundry and sold in the black market [7]; defective or dismissed ICs may be delivered as good ones [8]; IP core licenses may be violated and IP cores may be overused [9], [10]; designs may be maliciously modified to insert stealthy unwanted functionalities in the final product, also known as Hardware Trojan Horses (HTHs) [11].

A novel menace for microprocessor-based systems are *microarchitectural attacks* [12] [13]. These attacks are a sub-class of *Side-Channel Attacks (SCA)*, i.e., attacks that aim at leaking unauthorized information from the system by analysing timing information, power consumption [14], thermal footprint [15] or electromagnetic emanation [16]. On the other hand, microarchitectural attacks do not require the attacker to have physical access to the system under attack. Indeed, such attacks only rely on the observation of the timing behavior of the system while running sensitive applications. The basic idea behind this family of attacks is that since computer architectures are optimized w.r.t. processing speed there is a strong correlation between processed data, memory accesses and execution times: such correlation may represent an exploitable side channel in case the attacked and the attacker processes share the same cache space [12]. A well-known example of microarchitectural attack is *Meltdown* [17], where the attacker exploits out-of-order execution to break address space isolation without exploiting any software bug. Therefore, by exploiting Meltdown an attacker allows his/her own program to access the memory (and thus also secrets) of other programs and of the Operating System (OS).

In the past, hardware security was not considered an issue by the space industry because most attacks, e.g., fault attacks, required the attacker to have physical access to the attacked system, which indeed is not the case of space applications. Nevertheless, attacks based on MSCAs and HTHs may be successfully carried out by without any physical access (at the time of the attack) to the system. Indeed, once implanted, a HTH can be activated remotely through the execution of a trigger program as well as a MSCA can be carried out by forcing the execution of a piece of malicious software running in parallel with the attacked program. Therefore, hardware security issues should nowadays be tackled also when considering a digital design meant for a space application.

Microprocessors are present in On-Board Data Handling systems and On-Board Data Processing systems, both has hard cores in ASICs/SoCs and as soft cores in FPGAs. Microprocessors are key components to run the algorithms required for the Attitude Determination and Control System (ADCS) and execute ground-control or algorithm-based Command & Data Handling (C&DH) [18]. The orientation is critical for the satellite's mission objective (e.g. a wrong orientation would make communication with the ground station not possible). The execution of C&DH is critical too, given the master/agent relationship of ground stations to satellites [18].

In this perspective paper, we discuss the security issues related to the adoption of the RISC-V platform in space applications. We first present the security features that are natively offered by the RISC-V architecture and we present their limitations in preventing and alleviating the effect of hardware security-related attacks. Indeed, specifically looking at hardware security, we highlight how Hardware Trojan horses (HTHs) and Software Threats, with particular emphasis to Microarchitectural Side-Channel Attacks (MSCAs), may damage the overall system functioning by either altering its nominal behavior or stealing secret information. Finally, we overview the existing solutions coming from research that aim at further securing the adoption of RISC-V processors also presenting and discussing issues that have not yet been addressed. This paper is meant to help technicians in finding the best architecture configuration to meet both performance and security requirements. At the same time this work can represent a starting point for young researchers and students by providing a discussion of the state-of-the-art solutions to secure RISC-V as well as an overview of the open problems.

The remainder of this paper is organized as follows: Section II describes possible threats in space systems and their impact on different space missions; Section III discusses the main security-related features provided by the RISC-V architecture and their limitation; Finally, Section IV draws conclusions, highlighting open issues.

II. Security in Space Data Systems

In the space embedded systems domain we are now witnessing a novel trend. While "classical" processing ASICs keep suffering of the usual, widening, performance gap between 'terrestrial' processors and space grade ones (as discussed in [19]), on the counter recent developments especially in SRAM-based FPGAs have brought terrestrial 'edge' technologies for space use, providing a possible quantum leap for processing in space. Especially for payload data processing tasks, these devices will likely replace any dedicated application-specific standard product (ASSP) and call for availability of sophisticated processor IP cores for specific applications.

In the following we review the main security issues related to space missions highlighted by the Consultative Committee for Space Data Systems (CCSDS) in [20].

A. Threats

1) Intentional Data Corruption: Data may be intentionally altered by an attacker at its source (e.g. sensor jamming) by an attacking satellite operating nearby the attacked one [18]. Another possibility is that data is altered during its transmission from the instrument to the ground system, e.g., by the execution of malicious software. Along with the corruption of information coming from the instruments, intentional data corruption could potentially lead to a catastrophic loss of the system, e.g. if, at the reception of a command, no action or a wrong action is taken by the spacecraft. For example, if a navigational maneuvering command is altered, the spacecraft may eventually enter an unusable orbit and miss the encounter with a target, or even be destroyed.

2) Software Threats: Coding errors may happen and may cause security problems [20]. Furthermore, the system operator may configure the system incorrectly, leading to security weakness. On the other hand, the programmer may introduce logic or implementation errors that may lead to system vulnerabilities. Finally, external agents may try to use vulnerabilities to inject software or change configurations. The effect can be data loss, loss of spacecraft control, unauthorized spacecraft control, or mission failure.

Software Threats (STs) may also come from the exploitation of hardware- and architecture-level vulnerabilities. For instance, high-performance processors may employ speculation and out-of-order execution. These features can be maliciously used to access protected processor memory locations, opening the way to so-called Microarchitectural Side-Channel Attacks (MSCAs) [12]. This type of SCAs is particularly critical (also for space applications) because it does not require physical access to the system under attack, relying on the monitoring of the features of interest for the attack itself, e.g., timing behavior, performance counters, of the attacked processor. Some examples of MSCA, are the *Meltdown* [17], *Spectre* [21] and *Foreshadow* [22], where the attacker exploits out-of-order and speculative execution to break address space isolation without exploiting any software bug. Thus, these attacks allow the attacker to access unauthorized memory regions and steal information. As a result, a system built using trusted components may be successfully attacked.

3) Malicious hardware: Issues related to hardware trust arise from involvement of untrusted entities in the life cycle of the designed system, including untrusted IP providers, design team components, CAD tools and fabrication, test,

and distribution facilities [23]. It is possible that malicious HW finds its way in the system when COTS components are employed. These components usually do not have a trusted supply chain and are deployed to mass markets. As a matter of fact, the supply chain for COTS components consist of many (potentially untrusted) entities [23].

An undesired addition or modification to existing circuit elements, done in order to apply malicious activity is called Hardware Trojan (HT). HTs can change the functionality of a circuit, reduce its reliability or leak valuable information [23]. It has been known that *software-exploitable HTs* may be integrated in Microprocessor. Thanks to an HT, attackers may be able to execute their own malicious software, to modify the running software or to acquire root privileges [24], [25], [26]; or they may be able to enter in supervisor mode thanks to a backdoor activated via software [27]. Finally, it has to be mentioned that the most recent attack vector is now represented the employed CAD tools. Indeed, it has been demonstrated that HTHs may be introduced in the designed circuit by the tools employed for high-level and logic synthesis, for place-and-rout and for bitstream generation [28], [29].

B. Impact of threats on different space missions

The impact of security-related threats highly depends on the mission orbit and on the type of mission the system is meant for. In the following we discuss these two points.

1) Mission orbits: On the one hand, Geostationary Orbit (GEO) satellites are more secure than Low-Earth Orbit (LEO) missions because they require ground stations with a higher transmission power and larger antennas. This, of course, make communication with the satellite more expensive, thus limiting the number of possible attackers. On the other hand, GEO satellites are more vulnerable than LEO ones because they are continuously visible in specific geographical areas, while LEO satellites are visible only for specific time intervals from the ground station. Moreover, constellations of satellites increase the vulnerability of the whole space system, because in this case there are several satellites in orbit, increasing the visibility window from a specific point on Earth. Finally, deep-space/interplanetary missions require even larger antennas and higher power compared to a GEO satellite to attack [20].

2) Type of missions: The type of mission carried out by the space system, as well as its safety-/mission-criticality, highly influences the impact and likelihood of security-related threats. In the following we discuss several types of space missions and associated security issues.

Earth Observation Satellites can be either scientific or a critical asset of governments (e.g. meteorological forecasting applications). In the latter case the most likely threat is unauthorized access, which is typically countered with encryption of commands and mission data and authenticated commands. Moreover, malicious hardware is a possible threat that can be mitigated with analysis of the hardware functionalities, or careful selection of a trusted supplier. Data corruption may also be an issue, providing wrong information to the final user.

Communication Satellites are meant to provide a service with a usually very high availability. For example, in the case of a GEO telecommunication satellite the whole space system is expected to provide the service 99.9% of the time [30]. Therefore, these type of satellites are particularly vulnerable: indeed, even a slight reduction of the availability may have a huge impact on the service. An example of attack to a communication satellite is reported in [18]: in 2003, the 12 satellites of Telestar have been attacked using an uplink station that sent contradictory frequency to the satellites, which overrode the signal, effectively blocking television programming.

Navigation Satellites are often dual-use satellites, and often they belong to critical government infrastructures. An example is the GPS, that is employed by individuals, companies and the military. Like communications satellites, the loss of navigation satellite systems would result in potential loss of life, safety of individuals, and critical infrastructure.

Science missions are usually not part of a critical infrastructure. Even if they are exposed to the same threats as other missions, the impact is much less critical, with a damage to the financial investment and to the reputation of the entities involved in the mission itself [20]. However, investments might be very high: for instance, the cost of the recently launched James Webb Space Telescope reached 10 billion USD [31].

Manned missions are of course the most critical ones under the safety point of view. Indeed, in this cases the effect of an attack causing the loss of a spacecraft may be catastrophic, with the possibility of issues for the lives of the crew.

III. Security considerations on RISC-V

We here present the RISC-V architecture and discuss the available security-related features; finally, we survey the existing limitations and some proposed security extension.

A. The RISC-V Instruction Set Architecture

RISC-V was originally developed by UC Berkeley to support computer architecture research and education oriented at hardware implementations [32]. The spread of an open and free ISA already enabled a vast field of research activities for terrestrial application (e.g. security, AI, etc.). Indeed, having access to proprietary architectures is expensive and it limits what can be done with a certain product or within a certain research activity. The availability of a software ecosystem supported by a large open community ignited an unprecedented amount of developments, with several announcements and/or releases of open-source implementations. The adoption of a popular free and open ISA can thus lead to shorter time to market and lower costs thanks to reuse.

The main feature of the RISC-V ISA specifications is its modularity, which allows to cover a large spectrum of applications and to increase software reuse (adding new instructions as optional extensions instead of releasing new versions of the whole ISA). RISC-V allows for both standard and non-standard extensions (defined outside the specifications). The user-level RISC-V ISA is defined as a base integer (I) ISA,

which must be present in any implementation, plus optional extensions to the base ISA. A subset of the integer base (E) can optionally be implemented when an implementation targets small 32-bit microcontrollers, with 16 general purpose registers instead of 32. The standard defines a "general" subset (G) and the set of extension required for general purpose computing systems.

B. Memory isolation

The main feature available at software level to preserve the security of a system is memory isolation (i.e. each process has its own address space, which isolates memory between programs) [33]. Isolation implies that even if the security of a partition is compromised, the attacker cannot breach the security of other partitions. The Linux kernel provides separation between kernel services and user-space applications. However, in this way the kernel represents a single-point of failure in terms of security. Therefore, it may be responsible of a high number of vulnerabilities, given its huge attack surface [34].

Memory isolation can be further improved by employing hypervisors to provide isolation between different instances of OSs running on multicore processors. Memory isolation is currently being investigated in space applications to improve safety behavior, i.e. to avoid that the failure of an application running on an OS may impact the execution of another application running in a different OS on the same processor [35]. Given the increasing complexity of software systems also in space applications, designers increasingly rely on 3rd-party software components, typically provided as software libraries. When the code of these libraries is open source, a security analysis and validation is possible. On the other hand, proprietary software instead generally comes as linkable (and non inspectable) object modules. Therefore, it is often required to enforce the separation of the various software components within the system [34].

C. RISC-V software stacks and privilege levels

The modular nature of RISC-V allows for different software stack implementations with different levels of security [3]. For instance, Figure 1 shows the stacks described in the RISC-V Privileged Specification [36]. On the left side of the figure, a simple stack implementation is shown. The application interacts via an Application Binary Interface (ABI) with the Application Execution Environment (AEE). In this simple case the ABI is composed of the implemented user-level ISA subset. At the center of the figure, multiple applications run and communicate via the ABI with the OS (the latter in this case providing the AEE). In turn, the OS communicates with a supervisor execution environment (SEE) via a Supervisor Binary Interface (SBI). An SBI comprises the user-level and supervisor-level ISA together with a set of SBI function calls. The SEE can be a simple bootloader and BIOS-style IO system on a low-end hardware platform, up to a virtual machine in a high-end server, or a thin translation layer over a host operating system in an architecture simulation environment. On the right side of the figure, RISC-V runs a virtual machine monitor configuration where multiple OSs run on top of a hypervisor. Each OS communicates via an SBI with the hypervisor, which in turn provides the SEE. The hypervisor communicates with the hypervisor execution environment (HEE) using a hypervisor binary interface (HBI) to isolate the hypervisor from the hardware platform.

Privilege levels are used to provide protection between different components of the described software stacks. The possible levels a thread may have are:

- The *machine mode (M-mode)* has the highest privileges and it is the only mandatory one. Code running in machine-mode has to be totally trusted, as it has access to the machine implementation.
- The *supervisor mode* has been added to provide isolation between the Operating System and the SEE.
- The *user-mode* is dedicated to the user applications.

The privilege levels are designed and checked such that each attempt to perform operations not permitted by the current mode will cause an exception to be raised.

Any specific RISC-V implementation can choose which modes to implement (except for the mandatory M-mode). This allows hardware architects to identify a suitable trade off between offered security level and implementation cost [36]. Typical solutions are:

- M: simple embedded systems where security is not a concern. This solution will provide no protection against incorrect or malicious application code.
- M, U: secured embedded systems
- M, S, U: for systems running a Unix-like OS. The supervisor mode is added to provide isolation between the OS and the SEE.

A user thread normally runs the application code in U-mode until a trap, e.g., a supervisor call or a timer interrupt, forces a switch to a trap handler, that usually runs in a more privileged mode. The trap handler will be executed and eventually the user thread will resume its execution at or after the original trapped instruction, again in U-mode.

D. RISC-V extensions for security

Beside different privilege modes, the RISC-V ISA provides additional extensions to increase security.

1) Physical Memory Protection: The M-mode supports an optional standard extension for Physical Memory Protection (PMP). The PMP extension defines control registers to specify access privileges (read, write, execute) for each physical memory region. Attempting to fetch an instruction from a PMP region that does not have execute permissions or to load from a physical memory location within a PMP region without read permissions raises an exception.

The use of the PMP provides several security features. For example, threads are prevented from modifying or even reading the data of shared libraries. Therefore, the memory region in which the shared library data reside can be set as execute only using PMP. Even if the PMP is mainly used to prevent threads running in lower privilege levels (e.g. U

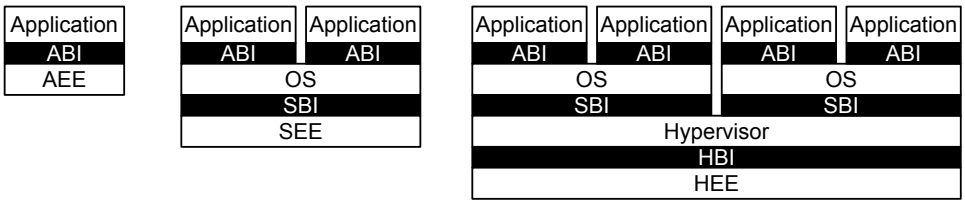

Figure 1: Examples of software stack implementations with different levels of security [36].

and S modes) from accessing privileged memory contents, the "lock" feature provides protection even when only the M-mode implemented. As a matter of fact, if a PMP entry is locked, write operations to the configuration register and the associated address registers of the entry are ignored. Indeed, PMP restrictions are valid also for the M-mode: locked PMP entries may only be unlocked with a system reset.

2) Cryptographic extension: The basic requirement for security in spacecrafts is the use the AES encryption standard to encrypt telemetry and telecommand signals [37].The implementation of a cryptographic extension increases performance and hides implementation details from software, reducing the attack surface [38]. However it is often preferable to implement encryption and decryption of telemetry and commands in a separated hardware component or board, as it is required that the keys are not accessible from software.

3) User-level interrupt extension (N): This extension allows isolation between interrupt handler having different privilege levels. Indeed, interrupt handlers can be executed at user-level, thus being unable to compromise the isolation model [34].

E. Limits of isolation and need for hardware solutions

Most attacks to isolation exploit physical access to the processor to perform Fault Injection (FI) [39]. In space this may be more complex than in applications like mobile and Internet of Things (IoT), although in principle still possible. For instance, in [39] it is shown how knowledge of power consumption for each instruction can be employed to skip an instruction during context switch, allowing access to PMP protected portions of memory. On the other hand, the previously presented attacks, i.e., MSCAs and HTHs, can be successfully carried out by the attacker without any physical access (at the time of the attack) to the system under attack. Indeed, HTHs can be implanted at design time by one of the many untrusted parties involved in the design. Once implanted, a HTH can "easily" be activated through the execution of a trigger program. Similarly, a MSCA is a piece of malicious software running in parallel with the attacked software. Therefore, MSCAs and HTHs have to be considered serious threats also in the space scenario, where it is extremely difficult for the attacker to have physical access to the system. While the security features natively integrated in the RISC-V architecture do not protect the system from MSCAs and HTHs, we argue that innovative security solutions (coming from research) should be integrated in the RISC-V architecture to make space missions adopting this microprocessor secure and trustable.

F. Security Architectures for RISC-V

Several system-level methodologies meant to protect systems based on Intel and ARM processors against both MSCAs and HTHs have been recently proposed. In particular, the new trend aims at providing a secure and trustable program execution over a system built with untrusted components. The main idea behind most of these approaches consists in the introduction Hardware Security Checkers (HSCs) able to monitor the fetching activity and, the processed data and the status of the Microprocessor for detecting potentially suspicious activities, e.g., HTHs and/or MSCAs.

The same effort has not yet been devoted to secure systems based on RISC-V processors. In [40] the authors proposed a HSC based on Bloom filters to protect the system against HTHs infesting the memories. At runtime, the HSC monitors the memory locations accessed by the microprocessor and the corresponding fetched instructions and based on this information it decides whether a HTH has been activated or not. The same authors proposed in [41] a similar solution where the target are HTHs infesting the microprocessor instead of the memory. Finally, probabilistic data structures are employed in [42] to detect the activation of MSCAs.

IV. Discussion and open issues

The growing interest in deploying commercial microprocessors in space applications as well as the rise of new hardware-oriented menaces, e.g., MSCAs and HTHs, make security (and hardware security, in particular) an open and severe issue. As we have discussed in the paper, RISC-V microprocessor already implements several security features. Indeed, Physical Memory Protection, Cryptograpy extension and User-level interrupts already protect a RISC-V based system against several attacks, e.g., data corruption and unauthorized access. On the other hand, the protection of RISC-V processors against MSCAs and HTHs is still an issue.

Apart from the protection against MSCAs and HTHs, we still see two main open issues related to the trusted adoption of RISC-V processors in space missions. The first issue is the lack of a trusted and robust tool chain: indeed, as it has already been discussed in the previous sections, it has been demonstrated that malicious modifications can be introduced in the designed system by the employed CAD tools. Therefore, the availability of trusted tools and of methodology to ensure the trustworthiness of the employed tools it is vital. The second main issue is related to the integration of third party intellectual property cores (3PIPs) within a RISC-V based

system. A trusted interaction between dedicated hardware accelerators and the main processor is fundamental to ensure security to the entire system. Therefore, methodologies to verify trusted behaviors and hardware-level mechanisms, like hardware-based firewalls, to isolate untrsuted components and limit their dangerousness should be investigated.

REFERENCES

[1] S. Esposito, C. Albanese, M. Alderighi, F. Casini, L. Giganti, M. L. Esposti, C. Monteleone, and M. Violante, "Cots-based high-performance computing for space applications," *IEEE Transactions on Nuclear Science*, vol. 62, no. 6, pp. 2687–2694, 2015.

[2] M. Pignol, "Cots-based applications in space avionics," in *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*. IEEE, 2010, pp. 1213–1219.

[3] S. Di Mascio, A. Menicucci, E. Gill, G. Furano, and C. Monteleone, "Leveraging the openness and modularity of risc-v in space," *Journal of Aerospace Information Systems*, vol. 16, no. 11, pp. 454–472, 2019. [Online]. Available: https://doi.org/10.2514/1.I010735

[4] S. Di Mascio, A. Menicucci, G. Furano, C. Monteleone, and M. Ottavi, "The case for risc-v in space," in *International Conference on Applications in Electronics Pervading Industry, Environment and Society*. Springer, 2018, pp. 319–325.

[5] DIGITIMES, "Trends in the global IC design service market," http://www.digitimes.com/news/a20120313RS400.html?chid=2.

[6] Mohammad Tehranipoor and Cliff Wang, *Introduction to Hardware Security and Trust*. Springer-Verlag New York, 2012.

[7] U. Guin, Ziqi Zhou, and A. Singh, "A novel design-for-security (dfs) architecture to prevent unauthorized ic overproduction," in *2017 IEEE 35th VLSI Test Symposium (VTS)*, 2017, pp. 1–6.

[8] U. Guin, K. Huang, D. DiMase, J. M. Carulli, M. Tehranipoor, and Y. Makris, "Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain," *Proc. IEEE*, vol. 102, no. 8, pp. 1207–1228, 2014.

[9] A. P. Donlin, P. Sundararajan, and B. J. New, "Method and system for secure exchange of ip cores," Aug. 2010, uS Patent 7,788,502.

[10] A. Carelli, C. A. Cristofanini, A. Vallero, C. Basile, P. Prinetto, and S. Di Carlo, "Securing bitstream integrity, confidentiality and authenticity in reconfigurable mobile heterogeneous systems," in *2018 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, 2018, pp. 1–6.

[11] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Design & Test of Computers*, vol. 27, no. 1, 2010.

[12] A. P. Fournaris, L. Pocero Fraile, and O. Koufopavlou, "Exploiting hardware vulnerabilities to attack embedded system devices: a survey of potent microarchitectural attacks," *Electronics*, vol. 6, no. 3, p. 52, 2017.

[13] R. Spreitzer, V. Moonsamy, T. Korak, and S. Mangard, "Systematic classification of side-channel attacks: A case study for mobile devices," *IEEE Communications Surveys Tutorials*, vol. 20, no. 1, pp. 465–488, 2018.

[14] C. Luo, Y. Fei, P. Luo, S. Mukherjee, and D. Kaeli, "Side-channel power analysis of a gpu aes implementation," in *2015 33rd IEEE International Conference on Computer Design (ICCD)*. IEEE, 2015, pp. 281–288.

[15] R. J. Masti, D. Rai, A. Ranganathan, C. Müller, L. Thiele, and S. Capkun, "Thermal covert channels on multi-core platforms," in *24th {USENIX} Security Symposium ({USENIX} Security 15)*, 2015, pp. 865–880.

[16] J. Longo, E. De Mulder, D. Page, and M. Tunstall, "Soc it to em: electromagnetic side-channel attacks on a complex system-on-chip," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2015, pp. 620–640.

[17] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, A. Fogh, J. Horn, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg, "Meltdown: Reading kernel memory from user space," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018.

[18] G. Falco, "When satellites attack: Satellite-to-satellite cyber attack, defense and resilience," in *ASCEND 2020*, 2020, p. 4014.

[19] G. Furano and A. Menicucci, *Roadmap for on-board processing and data handling systems in space*. Springer International Publishing, 8 2017, pp. 253–281.

[20] CCSDS, "Security threats against space missions," *Informational Report (CCSDS 350.1-G-2)*, 2015.

[21] P. Kocher, J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre attacks: Exploiting speculative execution," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 1–19.

[22] O. Weisse, J. Van Bulck, M. Minkin, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, R. Strackx, T. F. Wenisch, and Y. Yarom, "Foreshadow-NG: Breaking the virtual memory abstraction with transient out-of-order execution," *Technical report*, 2018.

[23] S. Bhunia and M. Tehranipoor, *Hardware security: a hands-on learning approach*. Morgan Kaufmann, 2018.

[24] Y. Jin, M. Maniatakos, and Y. Makris, "Exposing vulnerabilities of untrusted computing platforms," in *Proc. Int. Conf. Computer Design*, 2012, pp. 131–134.

[25] N. G. Tsoutsos and M. Maniatakos, "Fabrication attacks: Zero-overhead malicious modifications enabling modern microprocessor privilege escalation," *IEEE Trans. Emerging Topics in Computing*, vol. 2, no. 1, pp. 81–93, 2014.

[26] X. Wang, T. Mal-Sarkar, A. Krishna, S. Narasimhan, and S. Bhunia, "Software exploitable hardware trojans in embedded processor," in *2012 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*. IEEE, 2012, pp. 55–58.

[27] https://github.com/xoreaxeaxeax/rosenbridge.

[28] J. A. Roy, F. Koushanfar, and I. L. Markov, "Extended abstract: Circuit cad tools as a security threat," in *2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, 2008.

[29] M. Potkonjak, "Synthesis of trustable ics using untrusted cad tools," in *Proceedings of the 47th Design Automation Conference*, 2010, pp. 633–634.

[30] B. Kosinski and K. Dodson, "Key attributes to achieving >99.99 satellite availability," in *2018 IEEE International Reliability Physics Symposium (IRPS)*, March 2018, pp. 6A.3–1–6A.3–10.

[31] J. B. Pessoa, "Space age: Past, present and possible futures," *Journal of Aerospace Technology and Management*, vol. 13, 2021.

[32] A. Waterman, Y. Lee, D. Patterson, K. Asanovic, and V. I. U. level Isa, "The risc-v instruction set manual," *Volume I: User-Level ISA', version*, vol. 2, 2014.

[33] R. Shu, P. Wang, S. A. Gorski III, B. Andow, A. Nadkarni, L. Deshotels, J. Gionta, W. Enck, and X. Gu, "A study of security isolation techniques," *ACM Comput. Surv.*, vol. 49, no. 3, Oct. 2016. [Online]. Available: https://doi.org/10.1145/2988545

[34] C. Garlati and S. Pinto, "A clean slate approach to linux security risc-v enclaves," in *Proceedings of the Embedded World Conference, Nuremberg, Germany*, 2020.

[35] F. Gómez, M. Masmano, V. Nicolau, J. Andersson, J. Le Rhun, D. Trilla, F. Gallego, G. Cabo, and J. Abella Ferrer, "De-risc–dependable real-time infrastructure for safety-critical computer systems," *Ada User Journal*, vol. 41, no. 2, pp. 107–112, 2020.

[36] A. Waterman and K. Asanovic, "The risc-v instruction set manual volume ii: Privileged architecture document version 20190608-priv-msu-ratified," RISC-V Foundation, Tech. Rep., 2019.

[37] D. Lopez and E. Fraga, "Tm/tc encryption system," in *14th International Conference on Space Operations*, 2016, p. 2330.

[38] T. Lu, "A survey on risc-v security: Hardware and architecture," *ArXiv*, vol. abs/2107.04175, 2021.

[39] S. Nashimoto, D. Suzuki, R. Ueno, and N. Homma, "Bypassing isolated execution on risc-v with fault injection," Cryptology ePrint Archive, Report 2020/1193, 2020, https://ia.cr/2020/1193.

[40] A. Bolat, L. Cassano, P. Reviriego, O. Ergin, and M. Ottavi, "A microprocessor protection architecture against hardware trojans in memories," in *2020 15th Design & Technology of Integrated Systems in Nanoscale Era (DTIS)*. IEEE, 2020, pp. 1–6.

[41] A. Palumbo, L. Cassano, P. Reviriego, G. Bianchi, and M. Ottavi, "A lightweight security checking module to protect microprocessors against hardware trojan horses," in *2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*. IEEE, 2021, pp. 1–6.

[42] K. Arikan, A. Palumbo, L. Cassano, P. Reviriego, S. Pontarelli, G. Bianchi, O. Ergin, and M. Ottavi, "Processor security: Detecting microarchitectural attacks via count-min sketches," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pp. 1–14, 2022.