A Survey on FPGA Cybersecurity Design Strategies

ALEXANDRE PROULX, JEAN-YVES CHOUINARD, PAUL FORTIER, and AMINE MILED, Université Laval

This article presents a critical literature review on the security aspects of field-programmable gate array (FPGA) devices. FPGA devices present unique challenges to cybersecurity through their reconfigurable nature. The article also pays special attention to emerging system-on-chip (SoC) FPGA devices that incorporate a hard processing system (HPS) on the same die as the FPGA logic. While this incorporation reduces the need for vulnerable external signals, the HPS in SoC FPGA devices adds a level of complexity that is not present for stand-alone FPGA devices. This added complexity necessarily hands over the task of securing the device to developers. Even with standard security features in place, the HPS might still have unhindered access to the FPGA logic. A single software flaw could open up a breach that might allow an attacker to extract the FPGA's configuration data. A robust cybersecurity strategy is thus required for developers. As such, this work aims to provide the groundwork to build a solid threat-based cybersecurity design strategy that is specially adapted to SoC FPGA devices.

CCS Concepts: • **Security and privacy → Hardware attacks and countermeasures**; **Hardware security implementation**; **Embedded systems security**; • **Hardware** → *Reconfigurable logic and FPGAs;*

Additional Key Words and Phrases: Cybersecurity, SoC FPGA, FPGA, threat model

ACM Reference format:
Alexandre Proulx, Jean-Yves Chouinard, Paul Fortier, and Amine Miled. 2023. A Survey on FPGA Cybersecurity Design Strategies. *ACM Trans. Reconfigurable Technol. Syst.* 16, 2, Article 20 (March 2023), 33 pages.
https://doi.org/10.1145/3561515

1 INTRODUCTION

There exist two predominant **field-programmable gate array (FPGA)** technologies on the market today: **static random access memory (SRAM)**-based and flash-based technologies. Both of these technologies are featured in Figure 1. SRAM-based FPGA devices require an external **non-volatile memory (NVM)** to store their configuration information between power-offs due to the volatility of SRAM cells. In contrast, flash-based FPGA devices will retain their design indefinitely.

According to a 2019 market study by Gartner, the FPGA market consists of four significant manufacturers [21]. These are AMD-Xilinx at 51.1% of the market, Intel at 35.8%, Microsemi at 6.6%, and Lattice at 5.0%. The most prominent FPGA manufacturers, AMD-Xilinx and Intel, produce SRAM-based FPGA devices. Two other manufacturers, Microsemi and Lattice [80, 100], produce flash-based FPGA devices. Noting that SRAM-based FPGAs constitute the greatest proportion of the market today, the rest of this article will focus on this technology.

FPGA devices are particularly interesting to cybersecurity researchers since their hardware structure is defined mainly by the content of a memory array located within the device. This

Authors' address: A. Proulx, J.-Y. Chouinard, P. Fortier, and A. Miled, Department of Electrical and Computer Engineering, Université Laval, 2325 Rue de l'Université, Québec, QC G1V 0A6, Canada.

1936-7406/2023/03-ART20 $15.00
https://doi.org/10.1145/3561515

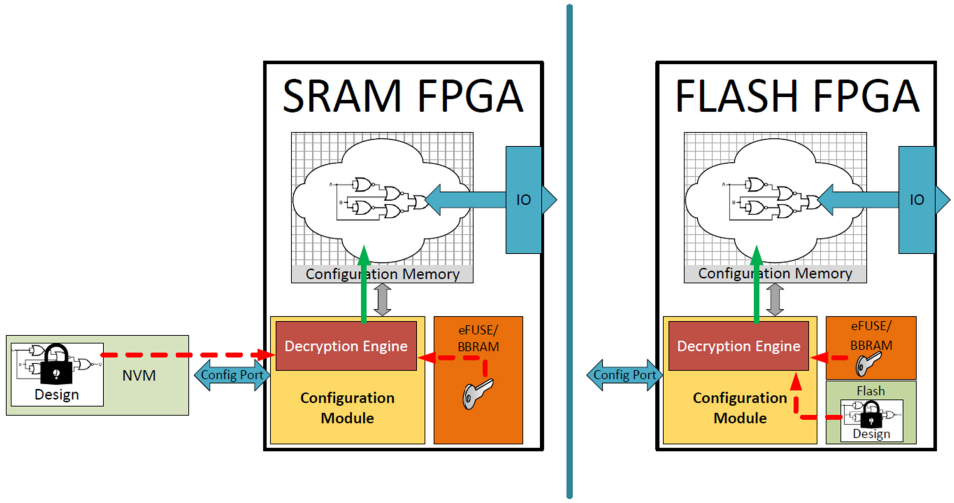


Fig. 1. SRAM and flash FPGA.

memory array, which we refer to as the FPGA's configuration memory, is loaded with a bit file, commonly known as the bitstream. The bitstream actively modifies switches and logic blocks, and interconnects within the device's configurable logic, also known as its fabric. Of the research works that deal with FPGA security, many seek to evaluate FPGA vulnerability to cyber attacks that compromise their bitstream. The bitstream is somewhat of a blueprint for the fabric of an FPGA. An attacker who successfully extracts a target's plaintext bitstream from an FPGA's configuration memory can reverse engineer this bitstream to either clone the intellectual property contained therein or add malicious functionalities back into the target. To protect their devices from attacks that specifically target the bitstream, FPGA manufacturers began adding complex encryption and authentication schemes [86, 127]. Although this adds a new layer of complexity for potential attackers, this article shows that multiple researchers have successfully compromised these security features.

More recently, FPGA manufacturers have begun incorporating elaborate processing systems on the same die as the FPGA. We commonly refer to these devices as **system-on-chip (SoC)** FPGA devices. By limiting the need for external signals, SoC FPGA devices provide an additional layer of security to communication paths between a processor and the FPGA. Although this improves the security of the processor-FPGA communication link, the complexity of the chip's design opens up these devices to new cybersecurity concerns. In cybersecurity terminology, the complexity of SoC FPGA devices expands the attack surface.

When discussing cybersecurity, we divide attacks into two general classes: active and passive. Attacks in both classes seek to break the confidentiality, integrity, and availability of a **target of exploitation (TOE)**. Active attacks seek to break a TOE by perturbing its normal function. A classic example of an active attack is a **fault injection (FI)** attack, where one will intentionally provoke the TOE to fall within an unintended state in an attempt to get it to leak restricted information [13, 68]. However, passive attacks seek to extract information from a TOE without interacting with its regular operation. **Side-channel attacks (SCAs)**, where one analyzes a TOE's external outputs and emissions to extract secret information, are classic examples of passive attacks [135].

By leveraging active and passive attacks, malicious actors can pose various threats to assets contained within FPGA and SoC FPGA devices. To secure a system against these malicious actors, system designers need to follow a methodology that thoroughly examines where security measures

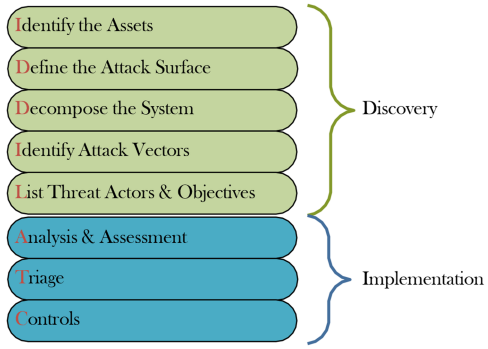


Fig. 2. IDDIL/ATC methodology [114].

(i.e., controls) need to be applied. Muckin and Fitch [114] propose one such methodology—the IDDIL/ATC methodology defined in Figure 2.

The IDDIL/ATC methodology is divided into two phases: discovery and implementation. In an attempt to keep the scope of this work general, we will primarily focus on the discovery phase and the activities contained therein. The system of choice will be a generic SoC FPGA with no specific design or operating environment. However, since basic security features such as encryption, authentication, and debug port disable are available on most recent SoC FPGA devices, these are assumed to be implemented. The final result will yield a generic threat model that outlines the attack surface, potential attack vectors, and threat actors of typical SoC FPGA devices. Since in-system execution of SoC FPGA devices is design dependent and will vary from one application to the next, we provide a review of applicable controls, functions, and interfaces without fully decomposing the system.

This article is organized as follows. Section 2 begins by discussing past works tackling the subject of FPGA cybersecurity. Next, FPGA and SoC FPGA assets and their attack surface are described in Section 3. We then describe techniques to identify threats in Section 4 and analyze multiple attack vectors that apply to FPGA and SoC FPGA devices in Section 5. Section 6 looks at security measures and controls one can implement in FPGA and SoC devices. Finally, Section 7 identifies future FPGA research and developments.

2 RELATED WORKS AND CONTRIBUTIONS

Several works have tackled the subject of FPGA cybersecurity in an approach that nears what this article seeks to achieve. Table 1 summarizes these works from the past 10 years.

Our work makes the following contributions:

- Provides a cybersecurity strategy designed to help developers identify those threats that apply to their systems and thus aid in applying appropriate countermeasures and controls.
- Provides a critical analysis of identified attack vectors and rates them according to their attack potential [147].
- Provides a threat model specially adapted to SoC FPGA devices and highlights the particularities of these devices and their applicable attack vectors.

3 GENERIC ATTACK SURFACE AND ITS ASSETS

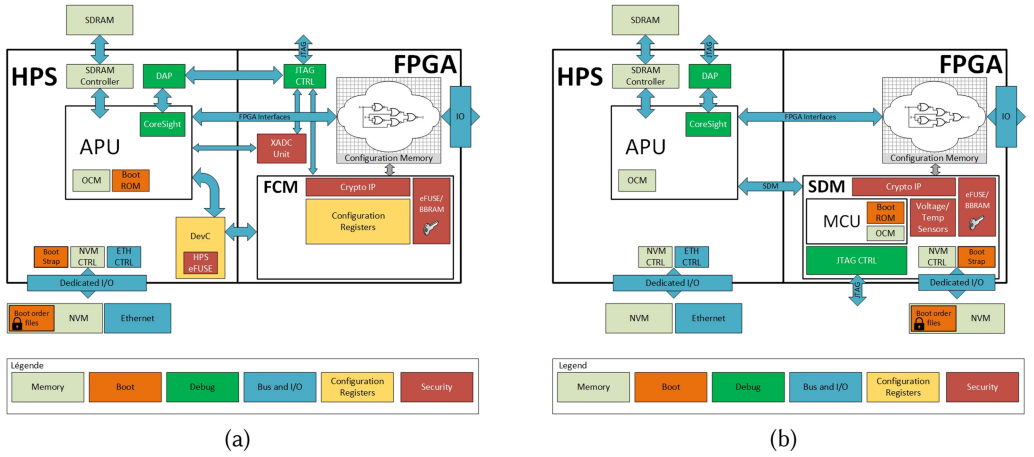
3.1 Assets

The first step of the IDDIL/ATC methodology is to identify those assets we wish to protect. Regarding FPGA devices, the primary asset of interest is the bitstream. Extending this principle to

Table 1. Related Works

Paper	Year	Relevant Contributions
Druyer et al. [36]	2015	Provides a comparison of the various controls available on prominent FPGA devices at that time. The Altera (Intel), Microsemi, and AMD-Xilinx devices are compared based on information obtained via various manufacturer white papers, datasheets, and other scientific papers.
Trimberger and McNeil [162]	2017	Provides a review of the security aspects of FPGA systems as they relate to software logic. Investigates how software security can be impacted by an FPGA design.
Benhani et al. [19]	2017	Provides a security review of the ARM TrustZone used within the AMD-Xilinx Zynq-7000 SoC FPGA. Hardware attacks and countermeasures directly related to SoC FPGA devices are presented.
Mirzargar and Stojilović [102]	2019	Provides a review of SCAs and covert channels that specifically apply to FPGA devices.
Zhang and Qu [174]	2019	Provides a review of FPGA-based systems while placing the emphasis on their supply chain. Known security issues and their corresponding defences are elaborated.
Duncan et al. [39]	2019	Provides a review of security issues and countermeasures that apply to the bitstreams of FPGA devices. A threat model for the life cycle of FPGA bitstreams is built.
Matas et al. [97]	2020	Provides a review of threats affecting FPGA devices within data centers and provides practical examples of attack and defense mechanisms and tools.
Turan and Verbauwhede [163]	2020	Provides a review of the security aspects surrounding the use of FPGA devices in cloud computing, supplying security assumptions and shortcomings.
Jin et al. [66]	2020	Presents a survey on the security aspects of FPGA devices in cloud applications. Builds a threat model based on possible threats to cloud FPGA users.
Huang et al. [56]	2020	Provides a survey on the use of machine learning in implementing controls against hardware Trojan attacks.
Duan et al. [37]	2021	Provides a literature review on security issues related to FPGA devices. Covers side-channel, FI, and covert channel attacks and their mitigations in FPGA and SoC FPGA applications.
Martínez-Rodríguez et al. [94]	2021	Provides a review of remote SCAs and their countermeasures.
Dessouky et al. [34]	2021	Provides a review of the challenges and security requirements for multi-tenancy FPGA-based cloud applications.
Rosero-Montalvo [138]	2021	Provides a review of the security weaknesses in both software and hardware FPGA development.
Sunkavilli et al. [149]	2021	Provides a survey on threats from FPGA electronic design automation tools.
Anandakumar et al. [11]	2021	Presents a survey on FPGA-based physically unclonable functions (PUFs) along with a detailed performance evaluation. Reports on known attacks and countermeasures related to PUFs.
Mahmoud et al. [92]	2022	Provides a threat model for CPUs, FPGAs, and GPUs, and examines side-channel, RowHammer, and FI attacks.
Pan and Mishra [125]	2022	Provides a survey of hardware vulnerability analysis methods that make use of machine learning techniques. The effectiveness of existing approaches and discussed, as well as open problems in the domain.

SoC FPGA devices, we also include any files required to boot the processing system. These may consist of firmware, bootloaders, baremetal applications, and **operating systems (OS)**. For both FPGA and SoC FPGA devices, we may also include decryption keys and any other form of data stored within these devices.



(a) (b)

Fig. 3. SoC FPGA layout: AMD-Xilinx Zynq-7000 (a) and Intel Stratix 10 (b).

3.2 Attack Surface

The IDDIL/ATC methodology's next step is defining the attack surface. The attack surface represents any components or elements that provide access to assets. A concrete example of how one might represent the attack surface is provided by taking up two competing SoC FPGA devices: the Intel Stratix 10 and the AMD-Xilinx Zynq-7000. This article's conclusions do not reflect on a particular manufacturer but instead attempt to identify vulnerabilities common to all FPGA or SoC devices. Furthermore, we should note here that during the writing of this article, Intel and AMD-Xilinx were producing the newer and more advanced Intel Agilex and AMD-Xilinx Ultrascale SoC FPGA, respectively. These SoC FPGA devices were not considered and could present various improvements not accounted for here.

3.2.1 Device Decomposition. When attempting to define the attack surface of a device, a diagram identifying the layout of critical sub-components is useful. Figures 3(a) and 3(b) show such diagrams for the Zynq-7000 and the Stratix 10 SoC FPGA devices, respectively. These layout diagrams note the various interactions and interconnect between individual sub-components.

Figure 3 highlights the function of the sub-components of each device as either a memory device or controller, playing a significant role in the boot process, part of the debug infrastructure, a bus, I/O, or bus controller, as containing important configuration registers, or as related to the security of the SoC FPGA.

The first distinctive difference between the two showcased SoC FPGA devices is their booting sequence. The Zynq-7000's initial boot sequence is wholly managed within the **hard processing system (HPS)**, whereas the Stratix 10 features a dedicated microcontroller unit within the **secure device manager (SDM)** to serve this purpose. A second distinctive difference is the pathways that lead up to each FPGA's configuration memory. In the case of the Zynq-7000, the configuration memory may be reached either by the application processing unit going through the device controller or directly via the JTAG interface. These paths lead through the FPGA configuration module, a state machine that manages FPGA configurations. Turning our attention to the Stratix 10, we find that all external interactions with the configuration memory are managed directly via the SDM.

Both the Zynq-7000 and the Stratix 10 feature ARM cores within their HPS. These cores have standard ARM infrastructure, including on-chip memory, access to NVM, ethernet interfaces,

synchronous dynamic random access memory (SDRAM), and their debug access port. Furthermore, both the SoC FPGA devices incorporate the ARM TrustZone technology [7, 62]. The ARM TrustZone technology allows resources within the FPGA fabric to be divided into two worlds: secure and non-secure. This feature is important for isolating safety-critical components and functions.

3.2.2 Data Flow Diagrams. Next, data flow diagrams can help outline the life cycle of assets. Figure 4 shows a data flow diagram from the development stage of SoC FPGA assets to the execution of an OS.

In drawing the data flow diagram, we assumed three basic security features were in place. First, we assumed assets were encrypted after development. The Intel Stratix 10 and the AMD-Xilinx Zynq-7000 use the **advanced encryption standard (AES)**-256 for this purpose. The Stratix 10 uses AES counter mode, whereas the Zynq-7000 uses the AES cypher block chaining mode. In both cases, we can store the secret keys for both devices within eFuses or **battery-backed random access memory (BBRAM)** [86, 129]. Second, we assumed authentication was in place for all assets before execution. Both the Intel Stratix 10 and the AMD-Xilinx Zynq-7000 allow for authentication. The Stratix-10 authenticates with an **elliptic curve digital signature algorithm (ECDSA)** via a **secure hash algorithm (SHA)**-256 or SHA-384 hash of the ECDSA public key stored within either eFuses or BBRAM [55]. The Zynq-7000, however, allows for Rivest, Shamir, and Adleman RSA authentication via a SHA-256 hash of the RSA-2048 public key and a subsequent secondary authentication via keyed-**hash message authentication code (HMAC)** [128]. HMAC is a message authentication code using a cryptographic hash function and a secret key. This secret HMAC key is stored within the encrypted boot files. The SHA-256 hash of the public key used for RSA authentication is stored within HPS eFuses. The final basic security feature involves debugging port disabling. The Stratix 10 and the Zynq-7000 allow developers to permanently disable the **Joint Test Action Group (JTAG)** interface used for programming and debugging via eFuse.

The following paragraphs will highlight each stage identified in Figure 4:

(a) In the development stage, developers and any third-party IP integrated within the design have access to assets. Once deployed, assets are placed within a memory space accessible directly via an NVM interface or remotely via an ethernet link.
(b) In this example, the **first-stage bootloader (FSBL)** is selected to be the first external partition loaded by the bootROM. Within the Zynq-7000, the bootROM executes on the application processing unit inside the HPS and will guide the boot process until the FSBL takes control. The bootROM is thus responsible for loading, decrypting, and authenticating the FSBL within the Zynq-7000. Within the Stratix 10, the bootROM executes inside the SDM and will hand over control to the SDM firmware after loading, decrypting, and authenticating the latter. The SDM firmware is then responsible for performing the same process for the FSBL.
(c) For either device, the bitstream is loaded either by the FSBL or **second-stage bootloader (SSBL)**. Of noteworthiness, with the Stratix 10, one can also configure the SDM firmware to load the bitstream within the FPGA's configuration memory before the FSBL [61]. We note in this stage that the SSBL and the OS are placed within the SDRAM for execution. The SDRAM is necessary since, in both the Zynq-7000 and Stratix 10, the on-chip memory is limited to 256 kB of memory, whereas SSBLs and OS are usually in the orders of several megabytes.
(d) The SSBL is a more robust bootloader than the FSBL and will feature drivers required for setting up the OS environment. The SSBL that typically loads Linux on ARM-based systems

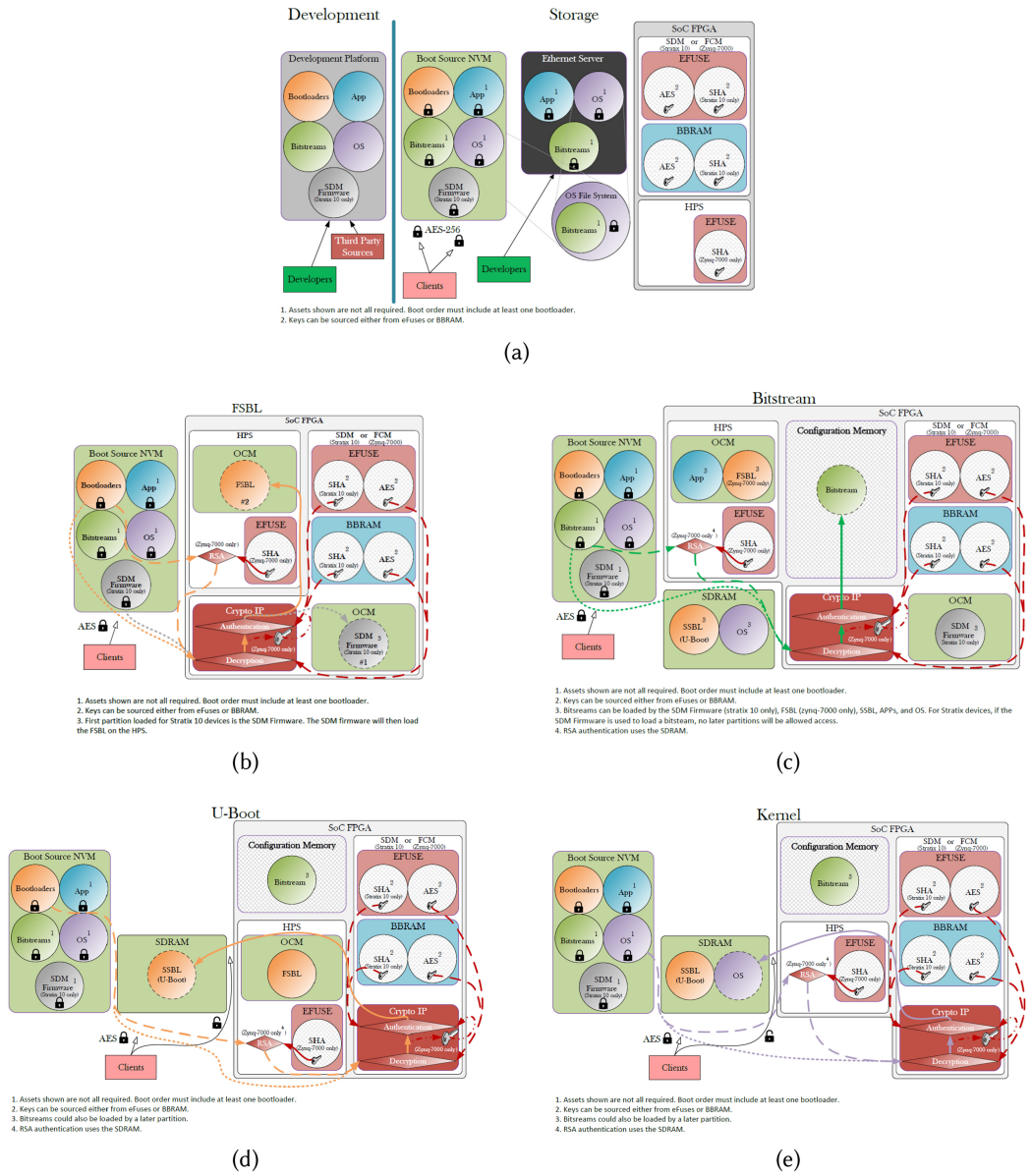


Fig. 4. Data flow diagram: Development (a), FSBL (b), Bitstream (c), U-Boot (d), and OS (e).

such as the Zynq-7000 and the Stratix 10 goes by U-Boot. For the Zynq-7000 and the Stratix 10, the FSBL is responsible for the decryption and authentication of the SSBL.

(e) Just like the FSBL was responsible for the decryption and authentication of the SSBL, the SSBL is responsible for completing the same process for the OS. Finally, the OS is placed within the SDRAM for execution.

The attack surface of common SoC FPGA assets has now been identified for its asset's life cycle from development to integration. The next logical step would be to define the attack surface during in-system execution. However, a valuable aspect of SoC FPGA devices is their ability to

be adapted to many interfaces and protocols based on their application. For example, within the HPS domain, the Zynq-7000 and the Stratix 10 feature dozens of HPS memory controllers, system modules, and interface peripherals ready to be implemented within any design. Furthermore, developers can implement countless interfaces within the FPGA domain, ranging from a customized ethernet interface to a simple universal asynchronous receiver transmitter controller. Understanding this, a data flow diagram for in-system execution will be design dependent and include all active interfaces within a said design.

4 IDENTIFYING AND ASSESSING THREATS

An essential step of a solid cybersecurity strategy is identifying threats to a given system. Knowledge of these threats will ensure that we can optimize resources allocated to applying controls to tackle those threats that pose the greater risk. This step needs to look at both applicable attacks and known vulnerabilities. Although these two notions are interrelated, the approach taken in identifying and assessing each is different.

4.1 Attack Vectors

Identifying attack vectors necessitates in-depth knowledge of tools, techniques, and methodologies that attackers can employ. Since cybersecurity researchers are consistently working at identifying such approaches on various devices, a great starting point is to perform a thorough literature review of their works. In this respect, Section 5 reviews the literature to capture attacks relevant to FPGA and SoC FPGA devices.

To help identify those attack vectors that pose a greater risk to a given application, a qualitative method to rate attack vectors based on attack potential [147] is proposed. Attack potential makes use of several criteria to rate attack vectors. These criteria include time constraints, the level of expertise and the knowledge of the TOE required, the level of access to the TOE or the window of opportunity, and finally the complexity of the equipment necessary to complete the attack. Table 2 presents these criteria and provides the weight distribution for the attack's identification and exploitation stages.

The identification stage is the stage of an attack where the effort required to create an attack and demonstrate its application to a particular TOE is compiled. The identification stage includes any action necessary to build and set up test equipment. In the exploitation stage of the attack, we apply the attack demonstrated in the identification stage while considering any difficulties encountered in expanding the attack to retrieve valuable results.

4.2 Vulnerabilities

Identifying known vulnerabilities could take a similar approach to that of attack vectors; however, vulnerability databases such as the **National Institute of Standards and Technology (NIST)** Vulnerability Database [120] provide comprehensive lists of known vulnerabilities. Furthermore, the Mitre organization [106] catalogues **common vulnerabilities and exposures (CVE)**, including those found within the NIST Vulnerability Database. For example, using the Zynq-7000 and the Stratix 10, Table 3 lists applicable CVEs that we can find.

The vulnerability score given in Table 3 is provided through **Common Vulnerability Scoring System (CVSS)** version 3.1 [43] from FIRST.Org Inc. (FIRST). The CVSS allows for a base, temporal, and environmental metric to be considered in the calculated score. The base metric designates those characteristics that do not change with time or across environments. Temporal metrics, however, touch on characteristics that may change with time but remains constant across environments. These characteristics could include the existence of tools to exploit the

Table 2. Attack Potential Criteria and Weight Distribution for the Identification and Exploitation Stages [147]

Factors	Identification (ID)	Exploitation (EXP)
Time		
<1 week	2	4
<1 month	3	6
>1 month	5	8
>4 months	6	10
Not practical	*	*
Expertise		
Layman	0	0
Proficient	2	2
Expert	5	4
Multiple expert	7	6
Knowledge of TOE		
Public	0	0
Restricted	2	2
Sensitive	4	3
Critical	6	5
Very critical	9	*
Not practical	*	*
Access to TOE		
<10 samples	0	0
<30 samples	1	2
<100 samples	2	4
>100 samples	3	6
Not practical	*	*
Equipment		
None	0	0
Standard	1	2
Specialized	3	4
Bespoke	5	6
Multiple bespoke	7	8

Table 3. Publicly Disclosed Vulnerabilities of the Stratix 10 and Zynq-7000

CVE Number	Device	Vulnerability	Vulnerability Score
CVE-2020-8737 [104]	Stratix 10	Improper buffer restriction	6.8 medium
CVE-2020-12312 [103]	Stratix 10	Improper buffer restriction	6.8 medium
CVE-2021-27208 [105]	Zynq-7000	Classic buffer overflow	6.8 medium
CVE-2021-44850 [107]	Zynq-7000	Classic buffer overflow	6.8 medium
CVE-2022-23822 [108]	Zynq-7000	Incorrect authorization	6.8 medium

vulnerability or the development of patches to fix it. Last, environmental metrics consider factors from the organizational infrastructure to adjust the overall CVSS score. These factors could include controls put in place or the effect a loss of confidentiality, integrity, or availability would have on the organization.

The CVSS scores in Table 3 feature the numerical values assigned to each respective CVE by NIST [120]. In all vulnerabilities featured, NIST only considers the base score. The omission of the other metrics makes sense for the application-specific environmental factors. However, regarding temporal metrics, the manufacturer fixes provided for CVE-2020-8737, CVE-2020-12312 [59], and CVE-2022-23822 [10], and the detailed exploitation summary available for CVE-2021-27208 and CVE-2021-44850 [137], would impact the calculated score. Last, all calculated scores are the same since, in all circumstances, the impact on the device's confidentiality, integrity, and availability is high. At the same time, the attack itself is purely exploitable via a physical attack. Vulnerabilities will generally earn a higher CVSS score when an attacker can achieve an exploit over a network.

5 LITERATURE REVIEW OF FPGA ATTACK VECTORS

In this section, we review the multiple attack vectors that can affect FPGA and SoC FPGA devices. We begin each discussion on a given attack vector by first providing a brief background on the attack vector, followed by a literature review of researchers who have applied these techniques directly to FPGA devices.

Moving forward, we recall the primary asset of FPGA devices, the bitstream, to realize that all of the attack vectors have something in common: whichever way an attacker successfully extracts a plaintext bitstream from the FPGA, the need to reverse engineer this bitstream remains. Therefore, as a starting point, we review bitstream reverse engineering tools and techniques found in the literature.

5.1 Bitstream Reverse Engineering

FPGA manufacturers facilitate the design process for their respective platforms with the help of **electronic design automation (EDA)** tools such as Vivado (AMD-Xilinx) [9], Quartus Prime (Intel) [60], Libero (Microsemi) [79], and Lattice Diamon (Lattice Semiconductor) [5].

All of these EDAs more or less follow a similar design flow. Developers start with **register-transfer level (RTL)** code. EDAs will synthesize this into a gate-level implementation or netlist and create a target-specific floorplan large enough to accommodate the design. From there, the EDA will place logic cells within this layout and route everything together. At the end of the process, the EDA's output will be a stream of bits destined to fill the FPGA's configuration memory and thus configure its logic cells and routing resources.

If one wishes to reverse engineer a bitstream from the bitstream back to the RTL code, one first must know the relationship between the bits contained within the bitstream and the gate-level implementation they represent. This relationship is manufacturer dependent and will vary between FPGA families and series. In this survey, we found that most of the work done on the subject thus far has been focused on AMD-Xilinx, Lattice Semiconductor, and Microsemi bitstreams [20, 30, 65, 71, 121, 130, 152, 175, 179]. As far as Intel is concerned, to the best of our knowledge, there have been no significant reverse engineering efforts on their FPGA bitstream. Given that it is a predominant FPGA manufacturer on the market today, one would expect similar advances to those seen on AMD-Xilinx FPGAs. However, we believe that since Intel, unlike AMD-Xilinx, provides very little information about the layout of their devices, researchers have so far focused elsewhere.

5.1.1 AMD-Xilinx. With AMD-Xilinx being one of the pioneers in the FPGA domain, we find that early research works aimed at reverse engineering bitstreams began here. In the early days, these were mainly focused on understanding the FPGA's internal structure to implement partial reconfiguration [40, 50, 82, 148, 173]. One of the first publications to analyze a bitstream from a cybersecurity perspective was presented by Ziener et al. [179], who sought to find a way of

identifying unlicensed proprietary cores within an FPGA from its bitstream. This work looked at the Virtex-II platform, whose user guide [6], much like previous AMD-Xilinx platforms, revealed a fair level of valuable details into the Virtex-II's logic cell structure and bitstream. This work successfully identified which packets within the bitstream represented lookup table content and used this content to identify specific unlicensed cores with high certainty.

Following these early attempts, Note and Rannaud [121] introduced the first publication to present a coherent algorithm to reverse engineer the bitstream back into its netlist. They made use of the now replaced AMD-Xilinx tool, *ncd2xdl*, which would translate the netlist circuit description file into a clear-text representation of the netlist, an AMD-**Xilinx design language (XDL)** file. The XDL file provided detailed information on the FPGA's internal state to identify configurable logic blocks and programmable interconnect points. With knowledge of the configurable logic blocks and programmable interconnect points present in a particular design from the XDL file and by the recurring structure of the FPGA, Note and Rannaud could relate an active site within the XDL file to its corresponding site within the bitstream. Using this established relationship, they created a program that automated the process of associating a particular portion of a bitstream to its corresponding XDL format. The next step performed by Note and Rannaud remains the basis of even the most recent bitstream reverse engineering tools: the creation of a database that holds a collection of associations between the location of bits within the bitstream and its corresponding netlist information. Using this database, one could step backward and retrieve critical portions of a plaintext bitstream in its XDL format. Although Note and Rannaud's work proved to be a significant advancement for the reverse engineering of the AMD-Xilinx bitstream, a considerable amount of information was still missing. With the intent of tackling this deficiency, Benz et al. [20] looked at AMD-Xilinx's netlist circit description report file (XDLRC). AMD-Xilinx uses the XDLRC file to describe the structure and resources of its various FPGA platforms. This improvement allowed Benz et al. to fully retrieve the FPGA architecture from the bitstream.

At the time of writing, a collaborative project by the name of F4PGA is attempting to create a universal open-source computer-aided design platform for FPGA development [116]. The project currently targets FPGA devices from two manufacturers: Lattice Semiconductor and AMD-Xilinx. However, the project aims at creating a platform to develop a much wider variety of FPGA architectures. Within F4PGA, at its lowest level, the logic hardware is described in a generic FPGA assembly (FASM) format devised for the project. F4PGA then has sub-projects to handle the transition from FASM to bitstreams from various FPGA manufacturers. These sub-projects gather the results from multiple individual bitstream generation runs to create device-specific databases. For AMD-Xilinx devices, this F4PGA sub-project goes by the name *Project X-Ray* [152]. Project X-Ray has successfully mapped out Artix-7 devices and is working on the remainder of AMD-Xilinx 7-series FPGAs. These same databases created for F4PGA bitstream generation can also convert a bitstream into the FASM format. From the FASM format, one can derive which features are enabled by the target bitstream [153].

5.1.2 Lattice Semiconductor and Microsemi. Bitstreams for devices from Lattice Semiconductor and Microsemi have also been reverse engineered recently. For Lattice Semiconductor, project Trellis [130] and project IceStorm [30] have successfully reverse engineered the bitstreams of the ECP5 and iCE40 devices, respectively. In the case of Microsemi, its ProASIC3, IGLOO2, and FUSION devices were successfully reverse engineered in 2021 by Kim et al. [71].

5.1.3 Deep Learning for Bitstream Reverse Engineering. Recently, Chen and Liu [27] showed how to recover function blocks from bitstreams using deep learning. They achieved this through a deep learning based object detection algorithm by first transforming the bitstreams of FPGA designs into images suitable for deep learning processing.

5.2 Side-Channel Attacks

SCAs represent one of the literature's first and most prevalent forms of physical attacks. They are passive attacks that seek to exploit weaknesses in the physical implementation of electronic devices. In an SCA, one analyzes changes in power supply [48, 109–111, 140, 166, 171, 172, 177], thermal signature [133], **electromagnetic (EM)** emanations [16, 63, 112, 170], photonic emanations [157], and timing [115] to extract secret information from a TOE. In the context of FPGA devices, attackers primarily use SCA against cryptographic engines to retrieve their secret keys; however, some use cases have demonstrated how side-channel emissions can also be used to recover input images and parameters from **convolutional neural networks (CNNs)** and binarized neural networks implemented within FPGA devices [166, 170].

In practice, SCAs normally begin with a series of measurements taken over the channel of choice. Since variation in a given channel due to a device's cryptographic operations is minimal, one normally takes the average of many of these measurements to reduce the effects of noise. Once a sufficiently large number of measurements are captured, an attacker can use several methods to recover the secret key. These methods include simple power analysis [73], differential power analysis [72, 74], template attacks [26], **correlation power analysis (CPA)** [23], and mutual information analysis [44].

5.2.1 SCAs on Cryptographic Algorithms. Turning our attention to practical examples of SCAs on FPGA devices, we find that one of the first such attack on an FPGA was attempted by Moradi et al. [109] in 2011. In this attack, they broke the AMD-Xilinx Virtex–II DES encryption by analyzing its power supply through differential power analysis techniques. Moradi et al. also performed similar attacks on AES encryption using CPA [110, 111] in 2012 and in 2013. In their 2013 attack, Moradi et al. targeted the Altera (Intel) Stratix II FPGA via its power supply. Using a digital oscilloscope and a custom programmer based on an ATmega256, they successfully retrieved the full AES-128 key in less than 3 hours.

As for other SCA mediums, Moradi and Schneider [112] also successfully mounted EM-based SCAs on AMD-Xilinx FPGA devices. Using a process similar to their previous CPA attack but replacing power readings with EM readings, they broke the AES-256 bitstream encryption of the 5, 6, and 7 series AMD-Xilinx FPGA devices. Comparing their attacks using EM and power side channels, they found that the positional accuracy due to the EM probe drove up the required number of traces. This trend was more significant as technology shrunk from 65 nm with the 5 series to 28 nm with the 7 series. Nevertheless, the non-intrusive approach to EM-based SCA remained a distinct advantage since it only requires one to place the EM probe close to the TOE. A related work by Iyer and Yilmaz [63] in 2019 proposes an adaptive acquisition protocol to help identify the optimal EM capture configuration. The protocol, tested on the AMD-Xilinx Artix-7 FPGA, was found to reduce the required acquisition time by a factor of close to 35.

Finally, recent SCA methods have turned to AI for significant improvements in the number of measurements required. This approach was presented by Ramezanpour et al. [134] in 2020, where they successfully extracted the key from an AES algorithm implemented within the fabric of an Artix-7 FPGA with less than 3,700 measurements. Their approach used unsupervised learning to extract the information required for the leakage model, thus allowing the attack to occur without any prior knowledge of the device. In another approach, Wang and Dubrova [164] demonstrate how deep learning using a single neural network classifier can recover the key from an AES algorithm implemented within an Artix-7. Their results showed they could recover the secret key with less than 430 measurements.

5.2.2 SCAs in Cloud Computing. Although all previously stated attacks require physical access to the TOE, the emerging trend of integrating FPGA devices within cloud computing instances is

raising new opportunities for attackers. Particularly, placing multiple tenants on a single device to share reconfigurable resources raises concerns over the leakage of sensitive information between isolated portions of the reconfigurable fabric through their power distribution network. With these applications in mind, recent works have shown that SCA techniques can be applied remotely [48, 133, 140, 177]. For instance, Gravellier et al. [48] made use of an AMD-Xilinx Zynq-7000 to show that they could infer the encryption key of both an AES instance running on a CPU and from a hardware implementation of the algorithm based in the fabric. Such capabilities could easily result in loss of confidentiality for unsuspecting cloud users.

5.2.3 SCAs on Neural Networks. Turning to other applications of SCAs, we find Wei et al. [166], who performed a power SCA to recover the pixel value of a CNN's input image. Their attack on the AMD-Xilinx Spartan-6 FPGA successfully recovered an image being processed in a classification task. In addition, Yu et al. [170] performed an EM-based SCA to retrieve the weight values of a binarized neural network. Using the AMD-Xilinx Zynq-7000 SoC FPGA, they showed that they could accurately recover the underlying model characteristics and develop a substitute model from these values.

5.2.4 SCAs on Physically Unclonable Functions. Yu et al. [171] in 2020 demonstrated how SCAs could further be used to classify the sequence of 1's and 0's from an FPGA's **physically unclonable function (PUF)**. Using an AMD-Xilinx Artix-7 FPGA as their target, they combined voltage-based SCA with deep learning to show that they could overcome PUF-based key provisioning and remote attestation measures.

5.2.5 SCAs on True Random Number Generators. Another application of SCAs comes with **true random number generators (TRNGs)**. The aim of SCAs applied to TRNGs is usually to determine the frequency of TRNGs implemented with **ring oscillators (ROs)**. Knowledge of this frequency can then be used to mount other attacks, such as FI attacks on the TRNG's output. One such approach using an EM-based SCA was presented by Bayon et al. [16] in 2013. Their attack revealed they could deduce the RO TRNG's frequency with high accuracy. Building on the work of Bayon et al., Yu et al. [172] in 2021 combined voltage-based SCA, deep learning, and a bitstream modification attack to extract the TRNG's output. Demonstrated on an AMD-Xilinx Artix-7 FPGA device, their attack resulted in a near-perfect accuracy.

5.3 FI Attacks

FI attacks are active attacks that seek to modify the behavior of the TOE. Ways of performing FIs include manipulating the TOE's temperature, supply voltages, or clock signals, or injecting external EM pulses, white light, laser, X-ray, or ion beams into the TOE [13, 68].

In experimental implementations of FI attacks, faults injected into a TOE cause transistors to switch abnormally. These abnormal transitions will lead to instruction skips or corrupted data values. The literature divides these fault attacks into three general sub-classes: algorithm modification, **differential fault analysis (DFA)**, and safe error [181]. Fault attacks that fall into the first sub-class, algorithm modification, will seek to skip or modify a critical instruction to circumvent a security measure. The second sub-class, DFA, is likely one of the most prevalent fault attacks. DFA seeks to inject faults in encryption and authentication mechanisms to retrieve their secret keys. The final sub-class, safe error, has a broader definition than its previous two sub-classes and features any fault attack that changes the expected behavior of a TOE.

Depending on the FI technique used, one can obtain differing results. Characteristics that define FI techniques include control over fault location and control over fault timing [68]. Both features will range from precise control to loose control to no control at all. For instance, injecting faults by

Table 4. Defining Characteristics and Equipment Requirement of Relevant FI Techniques

FI Technique	Papers	Required Equipment	Defining Characteristic
Voltage Glitches	[4, 24, 47, 77, 88, 91, 95, 122, 160, 161, 181, 182]	Standard lab equipment and low-cost custom FI circuits [122, 123]	Non-invasive, good temporal accuracy, and low positional accuracy
Clock Glitches	[2, 75, 95, 98, 132, 168, 181]	Standard lab equipment and low-cost custom FI circuits [123]	Non-invasive, high temporal accuracy, and low positional accuracy
Optical Attacks	[22, 24, 142, 143, 156]	Standard lab equipment and highly specialized equipment [52]	Invasive, high temporal accuracy, and high positional accuracy
Temperature Alterations	[95, 181]	Standard lab equipment [95]	Non-invasive, low temporal accuracy, and high positional accuracy
EMFIs	[17, 33, 90, 124, 126, 180]	Standard lab equipment and specialized equipment [1, 31]	Non-invasive, good temporal accuracy, and good positional accuracy

varying the device's supply voltage will offer no control over the fault's physical location, whereas injecting faults via laser will allow precise positional control. Table 4 summarizes the defining characteristics of each relevant FI technique.

5.3.1 FI Characterization Studies. Among the works conducted on FI, some researchers have sought to characterize the behavior of a given target due to FIs. Such characterizations were conducted for **electromagnetic fault injections (EMFIs)** by Zussa et al. [180] in 2014, again for EMFIs by Paquette et al. [126] in 2021, for voltage FI by O'Flynn [122] in 2016, and for laser FI by Selmke et al. [143] again in 2016.

5.3.2 FI Attacks on Cryptographic Algorithms. Other researchers have sought to execute specific attacks on FPGA devices. Most of these attacks demonstrated in the literature have sought to show the vulnerability of AES implementations on cryptographic modules within the FPGA [2, 22, 24, 33, 98, 124, 132, 143, 168, 181, 182]. For instance, in 2016, Selmke et al. [143] performed a laser FI on an AES core implemented within the AMD-Xilinx Spartan-6 FPGA device. Although the implementation featured a redundancy circuit, they could inject the same fault twice with a two-laser setup and thus induce exploitable faults into the circuit.

5.3.3 FI Attacks on SoC FPGA Devices. In 2016, Timmers and Spruyt [161] demonstrated an attack highly relevant to the discussion on SoC FPGA. Although this attack focuses explicitly on an ARM CPU, it also introduces the SoC FPGA attack vector. Their attack demonstrated that they could use a voltage fault attack to skip instructions processed by the ARM CPU on the Zynq-7000. Such capabilities raise important concerns for bitstream security. Suppose an attacker can gain control of the CPUs by skipping the authentication check. Access to the FPCA configuration module might be possible via the device controller module even though security features such as encryption and debug port disabling are in place.

5.3.4 FI Attacks in Cloud Computing. Researchers have also shown that they can launch FI attacks from hardware Trojans inserted within the fabric or from neighboring circuits in multi-tenant cloud computing platforms. This example was presented by Gnad et al. [47] in 2017, where they showed that an RO could be used as a form of voltage-based FI mechanism. They found that

the spontaneous current draws exerted by frequently activating the ROs would affect the device's normal operation. In their experiments, Gnad et al. validated the effects of such voltage-based FIs on the AMD-Xilinx Virtex 7, Kintext 7, and Zynq 7020 FPGA devices. In all three FPGA devices, the FI-induced errors ranged from complete system resets to minor malfunctions. Expanding on the works of Gnad et al., Krautter et al. [78] in 2018 formalized the hardware Trojan voltage-based FI attack as the *FPGAhammer*. Using FPGAhammer, they carried out a DFA on an AES implementation within the Intel Cyclone V SoC FPGA. Their attempt showed they could successfully inject timing faults via the RO-induced voltage drops. Their results showed that using approximately 35% to 45% of FPGA LUTs, they could recover 90% of the secret AES key.

Another remote FI technique was introduced by Alam et al. [4] in 2019. In their attacks, Alam et al. showed that dual-port RAMs in FPGA devices would allow for concurrent writes, which can result in memory collisions when opposing values are written to the same address simultaneously. These collisions cause transient shorts that can be exploited to increase the temperature of the FPGA. These temperature increases can induce timing violations and thus bit-flips in the FPGA device's configuration memory.

5.3.5 FI Attacks on PUFs. In 2015, Tajik et al. [156] demonstrated laser FIs on PUFs. In their experiment, the laser FIs were used to bypass specific countermeasures placed on PUFs to secure them against machine learning based attacks. In a second attack, they also showed how they could stop the ROs in a RO PUF, thus reducing the entropy of the numbers generated.

5.3.6 FI Attacks on TRNGs. Other researchers, such as Bayon et al. [17] in 2012 and Martin et al. [95] in 2015, have evaluated the impact of FIs on TRNGs implemented within FPGA devices. One of the latest such attacks was demonstrated by Madau et al. [90] in 2018. Using EMFI, they demonstrated how an EM pulse could affect the output of a TRNG implemented within an AMD-Xilinx Spartan-6 FPGA.

5.3.7 FI Attacks on Neural Networks. FI attacks are also possible on neural networks implemented within FPGA devices [83, 88, 178]. The most recent of these attacks, by Luo et al. [88] in 2021, used an AMD-Xilinx Zynq-7000 SoC FPGA to demonstrate how power glitching triggered through a specialized oscillating circuit was able to inject faults into a DNN on a neighboring portion of the fabric.

5.4 Probing Attacks

In a probing attack, an attacker attempts to monitor a die's internal signals directly. Several probing techniques exist to accomplish this. These are mainly divided between electrical and optical probing techniques [165]. Electrical probing techniques are those techniques that require direct contact with electrical paths within the die. However, optical probing techniques will either analyze photon emissions from transistors during switching activity or analyze light reflected on switching transistors after an external light source illuminates them. Probing attacks are, for the most part, invasive attacks; however, some non-invasive probing attacks can be found in the literature [157].

5.4.1 Electrical Probing. Electrical probing attacks are typically invasive, requiring attackers to physically decompose their target layer by layer to reverse engineer the electrical pathways within the chip [158]. This reverse engineering step will generally require the use of an optical microscope or a more expensive scanning electron microscope [145]. Once the target pathways are identified, we can use tools such as a focused ion beam system or a laser cutter to etch a hole and deposit the conducting material required for electrical probing.

Although FPGA and SoC FPGA devices are not immune to electrical probing attacks, examples of electrical probing attacks on FPGA are not prevalent in the literature. To provide a practical

example, however, we consider an attack demonstrated by Skorobogatov [146] in 2017. As his TOE, Skorobogatov targeted an 8-bit smartcard CPU core built with a 0.35-µm complementary metal-oxide-semiconductor process with three metal layers. Using a similar approach to what is described earlier, he extracted the entire memory space of the device successfully.

5.4.2 Optical Probing. Optical probing can be somewhat less invasive than electrical probing. In optical probing, one can also take advantage of the backside of the chip to access critical signals hidden deep within the chip. This approach is efficient on flip-chip packages, where the backside is readily accessible. In most cases, however, some chip decapsulation is still necessary to ensure the light can penetrate the target area.

As an example of optical probing on FPGA devices, we find Tajik et al. [157], who in 2017 successfully mounted an entirely non-invasive attack on the AMD-Xilinx Kintex-7 FPGA. This particular chip is available as a flip-chip package and thus provides direct access to the silicon substrate from its backside. Building on the previous work of Lorhke et al. [84] in 2016, without any modification to the device under test, by using a light source with a wavelength invisible to silicone, they could see through the Kintex's die. They were able to find and precisely map each bit of the bitstream as they exited the decryption engine. Using an internal clock of 33 MHz, they estimated a total acquisition time of 43 minutes for the whole bitstream. Furthermore, they estimated that the lab work required to complete the attack ranges from a few hours to a few days. As a further example of an optical probing attack, in 2018, using the same techniques as Tajik et al., Lohrke et al. [85] showed how optical probing could extract the full 256-bit AES key directly from the Zynq Ultrascale's BBRAM.

One of the latest optical probing attempts on FPGA was demonstrated by Krachenfels et al. [76] in 2019. They showed that it is possible to perform an attack similar to those presented by Tajik et al. and Lohrke et al. with a lower-cost laser FI setup. However, here, they noted a longer acquisition time.

Although the optical probing attack demonstrated by Tajik et al. exposes the vulnerability of flip-chip packages, we should note that most probing attacks will require some decapsulation effort in addition to the lengthy reverse engineering process. Furthermore, although Skorobogatov could quickly identify the data bus lines on the top metal layer, most secure chips will likely keep critical pathways deep within the sub-layers.

5.5 Hardware Trojans

During the attack surface identifications stage of the IDDIL/ATC methodology, we have seen how source code remains vulnerable to hardware Trojans while in their development stage. A seemingly unimposing piece of code inserted at this stage could translate into a significant vulnerability once deployed. Moreover, developers should take disproportionate measures to prevent their insertion.

5.5.1 Hardware Trojan Implementations. Looking at the literature, we find several works demonstrating potential hardware Trojan implementations. Among these implementations, we find one proposed by Chakraborty et al. [25] in 2013, where ROs are inserted into a design to reduce the device's lifetime; Ahmed et al. [3] in 2021 presented a Trojan that leaks out a target's AES key as it is being processed within an FPGA; and Ye et al. [169] in 2018 introduced a Trojan that can control the image classification process of a CNN implemented within an FPGA.

Other works, including Swierczynski et al. in 2015 [151] and in 2018 [150], and Ngo et al. [117] and Moraitis and Dubrova [113] in 2020, have investigated how adversaries could directly manipulate the bitstreams of cryptographic implementations in an effort to weaken them and thus make key recovery possible. Their results for differing cryptographic algorithms show that direct

bitstream manipulation can weaken cryptographic implementations without any reverse engineering requirement.

Some researchers have examined how Trojans could impact the ARM TrustZone technology found within many SoC FPGA devices [18, 49]. For instance, Benhani et al. [19] used the AMD-Xilinx Zynq-7000 SoC FPGA device to show that malicious modifications to the FPGA design could jeopardize isolation and segmentation put in place via TrustZone.

5.5.2 Hardware Trojan Insertions. Several points along the bitstream development chain have been identified as vulnerable to Trojan insertion techniques in the literature. Among the proposed approaches, we find Zhang et al. [176], who suggested in 2019 that one could insert Trojans through a malicious FPGA design suite. Another approach proposed by Ahmed et al. [3] in 2021 introduced a Trojan during the place-and-route step of bitstream generation. In their respective techniques, they could bypass all design check rules, thus ensuring that their Trojan remained undetected until their activation in the FPGA fabric.

5.6 Covert Channels

Covert channels show much resemblance to the previously discussed SCAs and are distinguished from them by whether one provoked the leakage of information or not. In an SCA, attackers exploit information that is accidentally leaked from the device, whereas in a covert channel, information is deliberately transferred from one device to another.

5.6.1 Thermal-Based Covert Channels. We find an example of a covert channel demonstrated on an FPGA device by Iakymchuk et al. [57] in 2011. They showed how two electrically isolated circuits on a single FPGA could communicate via a thermal-based covert channel. They established their covert channel with the use of specially designed ROs. They used a set of 20 ROs within the transmitter circuit to generate heat. A counter connected to a single RO within the receiver would detect any changes in the RO's frequency induced by variations in the die's temperature. Iakymchuk et al. showed how they could use this setup to transfer the secret key from an AES implementation to an unsecured portion of the FPGA at a rate of 0.5 bits per second. At this transmission rate, an error rate of 5% to 13% was observed. Other similar thermal-based covert channels have also been presented by Masti et al. [96] in 2015, by Bartolini et al. [15] in 2016, and by Tian and Szefer [159] in 2019.

5.6.2 Voltage-Based Covert Channels. In 2018, Nguyen [118] published a thesis where he introduced a voltage-based covert channel. This covert channel showed many similarities to the previously introduced thermal covert channels; however, whereas previous covert channels hid information in timing variations, he hid information in the voltage's amplitude. In 2019, improving on the work of Nguyen, Gnad et al. [46] presented a second voltage-based covert channel that introduced modulation into the system. They showed that by using less than 3% or 5% of the surface area of the AMD-Xilinx Kintex-7, they could transfer up to 8 Mbits per second via this covert channel while maintaining an error rate of 0.003%.

5.7 SoC FPGA Devices and Logical Attack Vectors

The complexity introduced by processing systems, peripheral interfaces, and even overengineered security features provides attackers with an extensive range of potential attack vectors. These attack vectors become increasingly relevant as heterogeneous systems are integrated to form SoC devices and network connectivity for embedded systems increases. Among the subjects already discussed, we mentioned how FI attacks on the processing system of a Zynq-7000 could impact the security of the fabric; however, this is just one of many such attacks targeting SoC FPGA devices and other logical attack vectors.

Table 5. Attack Potential Comparison of Attack Classes (ID, EXP)

Factor	Reverse Engineering	SCA	Probing Attack	Voltage FI	Clock FI	Optical FI	Temperature FI	Electromagnetic FI	Covert Channels	Hardware Trojans
Time	6, 4	3, 4	2, 4	3, 4	3, 4	3, 4	3, 4	3, 4	3, 4	3, 4
Experience	7, 2	7, 2	7, 4	7, 4	7, 4	7, 4	7, 4	7, 4	5, 2	5, 2
Knowledge	4, 0	6, 0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0	4, 3	4, 3
Access	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0
Equipment	1, 2	3, 4	7, 8	3, 4	3, 4	5, 6	3, 4	3, 4	1, 2	1, 2
Total	26	29	32	25	25	29	25	25	24	24

5.7.1 Buffer Overflow Attacks. We also presented several vulnerabilities of the Zynq-7000 and Stratix 10 in Section 4.2 [103–105, 107, 108]. From two of these vulnerabilities, CVE-2021-27208 [105] and CVE-2021-44850 [107], a severe exploit is possible on the Zynq-7000. Using these specific vulnerabilities, Schretlen [137], demonstrates specific exploits that can result in arbitrary code execution on the CPU.

5.7.2 Auto-Decryption Oracle. Another logical attack was demonstrated on an AMD-Xilinx 7-series FPGA by Ender [42] in 2020. Ender introduced several weaknesses of AMD-Xilinx 7-series FPGA, which, when put together, can allow for a previously encrypted bitstream to be read out 32 bits at a time in its plaintext form.

5.7.3 Breaking Secure Boot. A more elaborate scheme by Jacob et al. [64] in 2017 showed that a hardware Trojan inserted into the bitstream of a Zynq-7000 FPGA could perform data modifications within the system's external SDRAM. In their proof of concept attack, Jacob et al. decoyed a Trojan within a hardware module that resembled a cryptographic accelerator. They used this hardware Trojan to exploit a vulnerability in the secure boot process of the AMD-Xilinx Zynq-7000 that necessarily forced the device to execute a malicious kernel. After loading the U-Boot into SDRAM, they demonstrated that they could instruct the device to load a non-encrypted malicious kernel by modifying a few lines of this U-Boot's image. Furthermore, the device required no partition authentication before executing this malicious kernel.

5.8 Summary of Attack Vector Analysis

To conclude the review of FPGA attack vectors, we present a comparison of all previously discussed attacks. Table 5 shows the computation of all five factors of attack potential. SoC FPGA devices and logical attack vectors have been excluded from this comparison, as they tend to apply to specific circumstances and for particular devices. Instead, we created Table 6 to specify applicable targets and configurations that are prerequisites to the attack potential.

In assigning the score shown in Table 5 for bitstream reverse engineering, since a great deal of the work goes into the identification stage of the attack, we attributed more time, experience, and knowledge here. Once the relationship between the bits and the RTL code is known for a given device, exploiting a given bitstream is trivial. Furthermore, this attack will almost always apply as a subsequent stage to one of the other attack vectors. Thus, one should compound the rating for bitstream reverse engineering with the vector used to extract the bitstream.

Table 6. Attack Potential Comparison of Target-Specific Attacks (ID, EXP)

Factor	Schretlen, 2022 [137]	Ender et al., 2020 [42]	Jacob et al., 2017 [64]
Target	Zynq-7000 SoC FPGA	AMD-Xilinx 7-Series FPGA	Zynq-7000 SoC FPGA
Configuration	All Configurations	All Configurations	U-Boot and Linux Kernel
Time	5, 4	2, 4	3, 4
Experience	7, 2	7, 2	5, 2
Knowledge	6, 0	0, 0	4, 3
Access	0, 0	0, 0	0, 0
Equipment	1, 2	1, 2	1, 2
Total	27	18	24

Turning our attention to the score given to Trojan insertion, both for generic hardware Trojans in Table 5 and for the attack by Jacob et al. shown in Table 6, this score reflects a device where the bitstream has been encrypted. In such cases, the difficulty of the attack is primarily represented in the attempt to exploit the window of opportunity, which must be well guarded and restricted. This window is represented by the knowledge factor, where Common Criteria Methodology [147] defines a rating of 4 as sensitive information limited by strict need-to-know and specific contracts. In this respect, the insertion of a Trojan along the development chain would require a sensitive level of access. However, if the bitstream is unencrypted, the process is relatively simple.

When it comes to the attacks demonstrated by Schretlen [137] and the other by Ender [42], they represent complex attacks that require an in-depth understanding of the FPGA and SoC FPGA device. However, the main difference between the two attacks is that whereas Ender primarily connected the dots between manufacturer documentation and various known vulnerabilities, Schretlen extracted the sensitive and proprietary bootROM from the device and identified key vulnerabilities therein. Hence, this attack's identification phase gets an unusually high knowledge factor rating. Finally, we find that this attack represents a relatively low attack potential for the exploitation stage since the critical information is now public and a "plug-and-play" payload exists [137].

6 APPLYING CONTROLS

So far, we have introduced the attack surface of SoC FPGA devices, presented tools to identify and assess threats from various attacks and vulnerabilities, and then provided a thorough review of attack vectors that apply to FPGA and SoC FPGA devices. During this review of attack vectors, we used the tools introduced to assess the attack classes reviewed. With this information, the next logical step of the IDDIL/ATC methodology defined in Figure 2 is to use this assessment, along with the metrics assigned to known vulnerabilities and identify what controls need to be put in place.

The essential function of a control is to remove, counter, or mitigate a threat. Therefore, when seeking to apply a control, we need to identify which attack vector or vulnerability poses a more significant threat to the system. Thus, to identify (i.e., select) the proper controls, the previously identified threats should first be categorized. A model such as the STRIDE-LM model proposed by Muckin and Fitch [114] and depicted in Table 7 can be used for this step.

We can sort the threats using the STRIDE-LM model and ensure controls are applied optimally. Controls that fit into these categories will vary; some are manufacturer provided and can be applied by system designers, whereas researchers proposed others that might need to be implemented by FPGA manufacturers. Sections 6.1 and 6.2 provide a review of such controls as presented by FPGA manufacturers and the literature, respectively.

Table 7. Threat Categorization, Security Properties, and Controls [114]

STRIDE-LM	Threat	Property	Definition	Controls
S	Spoofing	Authentication	Impersonating someone or something	Authentication mechanism
T	Tampering	Integrity/access control	Modifying data or code	Crypto Hash, watermarking, PUF
R	Repudiation	Non-repudiation	Claiming to have not performed a specific function	Logging
I	Information disclosure	Confidentiality	Exposing information or data to unauthorized individuals or roles	Encryption, isolation
D	Denial of service	Availability	Deny or degrade service	Redundancy
E	Elevation of privilege	Authorization/ least privilege	Gain capabilities without proper authorization	Isolation and authentication
LM	Lateral movement	Segmentation/ least privilege	Expand influence post-compromise; often dependent on elevation of privilege	Segmentation and boundary enforcement

6.1 Manufacturer-Provided Controls

For many identified threats, manufacturer-provided security features will provide a reasonable level of protection. Comparing these features on Intel and AMD-Xilinx devices, we find multiple similarities [86, 127].

Intel and AMD-Xilinx's most crucial security feature implemented within FPGA devices is the AES encryption [119] used to encrypt their bitstreams and boot order files. Most FPGA attacks have focused on this particular security feature. Once one obtains the bitstream's plaintext format, the only obstacle between an attacker and a victim's sensitive design is the bitstream's complexity or security through obscurity. Although FPGA manufacturers go to great lengths to keep their bitstream structure secret, multiple sources have shown that mapping the bitstream to a netlist or even RTL code is not an impossible feat.

A secondary security feature implemented by both Intel and AMD-Xilinx is authentication. Authentication prevents an attacker from uploading malicious bitstream (or malicious partitions in the case of SoC FPGA devices). Manufacturers tend to use HMAC, RSA, ECDSA, or some combination of the three [14].

Last, FPGA manufacturers usually provide a set of eFuses on their devices to permanently alter certain functionalities. One can disable standard functionalities, including JTAG [58] access and bitstream readback. JTAG is helpful during the design and debugging of the system; however, it becomes a convenient point of entry for attackers once fielded. If a designer wishes to maintain post-deployment debug capabilities, he can disable its most intrusive function: its ability to read back a programmed bitstream. On the Stratix 10, Intel indicates that this feature is permanently disabled [62].

6.2 Security Measures from the Literature

In some applications, manufacturer-provided controls might not provide a solution that adequately responds to a given threat. For instance, emerging applications in cloud computing and

multi-tenant environments might pose threats that were not fully accounted for, or researchers might have uncovered new vulnerabilities in specific devices. We can turn to the literature in such cases, where multiple researchers have recommended innovative tools and techniques to help apply controls.

6.2.1 Bitstream Reverse Engineering. When it comes to securing bitstreams against reverse engineering, the first line of defense should be bitstream encryption. As such, efforts to improve cryptographic implementations within FPGA devices should be the main priority; however, in cases where bitstream encryption is not practical or where the device of interest has known vulnerabilities that affect its cryptographic implementation, one should seek techniques for bitstream obfuscations.

Hoque et al. [54] proposed one such bitstream obfuscation method in 2019. Their method uses unused LUTs to obfuscate critical and non-critical areas of the FPGA design to render their functions and structural properties indiscernible. Furthermore, Hoque et al. also implement redundancies to prevent attackers from attempting to uncover the system's functionalities via targeted, rule-based, and random tampering.

6.2.2 Side-Channel Attacks. The logical way to secure devices against SCAs is to reduce side-channel leakage. This reduction will generally take on two different forms: hiding and masking.

Hiding seeks to normalize channel information that attackers could use to recover secret information. For instance, in the case of timing attacks where execution time or other temporal references are used to extract secret information, controls will usually seek to remove the dependence on time by making all operations equally long [115]. When it comes to voltage-based SCA, Le Masle et al. [81] propose a countermeasure that monitors power consumption to keep the latter constant.

However, masking seeks to obscure intermediate values by injecting randomness via TRNGs to remove the dependence of side-channel leakage on the secret information we wish to protect [29, 38]. Although this method has been mainly applied to protect secret keys in cryptographic implementations, a recent work by Dubey et al. [38] in 2020 demonstrates how one can use masking to protect weight distribution in neural networks.

A uniquely FPGA-related countermeasure to SCA involves using partial reconfiguration to randomize lookup tables within the FPGA fabric. This approach was presented by Sasdrich et al. [139] in 2015, where they updated a new random S-Box configuration for every encryption sequence. Furthermore, noting the impact of placement and routing on side-channel leakage, other preventive countermeasures pay special attention to the placement of critical modules and the length of wires being routed [87, 89, 141].

Last, we also find other researchers who have focused on the development of tools and frameworks to evaluate side-channel emissions [41, 45, 67, 69, 70, 123].

6.2.3 FI Attacks. Moving on to FI attacks, the most common approach to secure against these attacks involves passive solutions such as redundancies and active solutions such as glitch detectors.

Redundancy is a clear way to prevent faults and can be implemented in several ways. First, a popular approach employs triple modular redundancy designs. These solutions have been studied extensively and implemented in most recent FPGA devices to secure sensitive pathways such as debug access circuits [8]. Another approach involves redundant algorithms, especially in cryptographic implementations [35, 93]. Similar to what was described for SCA, we also find applications for partial reconfiguration as countermeasures for FI attacks. This approach was presented by Mentens et al. [99] in 2008, where the location of the cryptographic engine is randomized to hinder

FI attacks. Regarding SoC FPGA, software redundancies should also be implemented for critical security functions such as authentication [161]. Bypassing a single check is hard, but bypassing two checks is much less likely.

Active controls seek to detect the effects of FIs within the FPGA. Such solutions were proposed by He et al. in 2016 [51, 52], and in 2017 [53]. He et al. proposed high-frequency RO watchdogs to detect laser FIs within the FPGA fabric. Similarly, Shen et al. [144] in 2019 used a delay sensing circuit within the FPGA for detection. Upon detection, they tried several response mechanisms that sought to delay the rising edge of the system's clock. In this manner, they would mitigate the transient's effect on the system. Other approaches that specifically attempt to identify and locate malicious tenants in multi-tenant application were proposed by Provelengios et al. [131] in 2019 and Mirzargar et al. [101] in 2020.

6.2.4 Probing Attacks. Where physical access is possible, an attacker with sufficient means will, in most cases, be able to extract information directly from a device via electrical or optical probing attacks. The best we can do is make it as hard as possible for an attacker to extract valuable information; a few ways of doing this exist.

Tajik et al. [155] propose a countermeasures to their optical probing attack. The countermeasure proposed is a PUF-based security monitor. PUFs rely on the physical characteristics of a given device to generate a unique signature or fingerprint. Tajik et al. propose using an RO PUF whose defining physical characteristics are known. Their experiments showed that attempts at optically probing the internal circuit would impact characteristics of the PUF in a way that could be successfully detected with high probability.

6.2.5 Hardware Trojans. Securing a device against hardware Trojans is mostly about knowing and understanding what composes your design. The best and only foolproof way of securing against hardware Trojans is to control the entire development chain. From this controlled development chain, one can then put authentication and integrity controls in place using secret keys, watermarks, or PUFs to ensure no changes are made to the design. However, this quickly becomes impractical as complexity increases, and there is a need to implement higher levels of abstraction through third-party building blocks. Hence, requirements for Trojan detection become apparent.

Most Trojan detection techniques found in the literature have focused on validating the authenticity of a given design based on defining characteristics. For instance, the work of Söll et al. [154] in 2014 used EM emissions to detect discrepancies between a legitimate design and its FPGA implementation. Similarly, more recent works, such as that of Danesh et al. [32] in 2021, have taken an approach akin to what is used in software security by stepping back into a design through the bitstream reverse engineering. After decomposing their designs, they can use various techniques to identify hidden malicious circuits. Another work of interest by Chithra et al. [28] in 2020 used machine learning to detect Trojans. They showed that they could detect Trojans based on temperature and voltage values obtained from different standard benchmarks of the AES encryption algorithm.

Furthermore, Krachenfels et al. [76] in 2021 proposed using laser-assisted optical probing to awaken dormant Trojans within an FPGA fabric. This technique could help uncover Trojans that have successfully bypassed the EDA's design check rules.

Other security measures against Trojans include isolation and segmentation to limit the impact of malicious hardware within the FPGA fabric. Taking the attack by Jacob et al. described in Section 5.7.3 as an example, if the HPS had configured ARM's TrustZone prior to programming the FPGA, then this attack would not have been possible. However, as described in Section 5.5, TrustZone is not infallible. As such, some researchers have begun looking at new techniques for trusted execution environments better adapted to FPGA devices [12, 136, 167]. For instance,

Ren et al. [136] proposed a scheme that uses remote attestation to validate hardware accelerators programmed within FPGA devices. They showed how their scheme could validate accelerators deployed on remote cloud infrastructures.

7 FUTURE FPGA RESEARCH AND DEVELOPMENT

Cybersecurity is a continually evolving field of research. New vulnerabilities arise as new systems are developed, and new unknown attack vectors are formed as new techniques are integrated. Consequently, identifying, prioritizing, and mitigating cybersecurity threats requires a thorough methodology to ensure systems are designed securely. This need is especially true regarding FPGA and SoC devices in the Internet of Things and operational technology, where these devices must be designed to operate independently for extended periods. In such circumstances, physical security and expedient security updates are not necessarily applicable. Furthermore, although traditional cybersecurity is well established in information systems, these methodologies do not necessarily translate very well to embedded systems.

Keeping to the particularities of FPGA and SoC FPGA devices, this article has reviewed methodologies and techniques for secure embedded system design. First, we showed how knowledge of the attack surface could be leveraged to provide the necessary input to subsequent phases of the cybersecurity analysis. Second, we performed a detailed literature review of applicable attacks and vulnerabilities. Third, we used attack potential and the CVSS score to assess their threat. Finally, we showed how controls could be selected to respond to identified threats by leveraging the STRIDE-LM model. Throughout this review, we have drawn from works focused on traditional cybersecurity and extended their approach to the world of embedded systems. Although our approach provides a basis from which to build, it is up to the scientific community, developers, and manufacturers to take up and improve on this work.

Speaking on improvements, since our primary focus has been on reviewing the works of past cybersecurity researchers, drafting a cybersecurity methodology, and gathering and testing the necessary tools, we have but touched the surface of what needs to be accomplished to define the cybersecurity approach fully. In the future, we note several areas of research and development that can help augment cybersecurity in FPGA and SoC FPGA devices:

- *Cybersecurity assessments*: Ideally, the assessments featured in Table 5 would be broken down to specific devices, targeting specific assets. As such, there is a need for further research to test and evaluate these attack vectors as they apply to specific devices. Manufacturers should accomplish this via rigorous penetration-testing experiments to help designers choose the right device for their application. A more productive approach, however, would be for manufacturers to deliver open designs that cybersecurity researchers can scrutinize, evaluate, and improve. Many leaps in cybersecurity are made by curious individuals who will stop at nothing to find a loophole in a system. These exposures allow users and developers to adapt their systems and ensure their sensitive information remains safe.
- *Applied AI*: Table 5 shows that hardware Trojans and covert channels share one of the weakest attack potentials. Although an apparent safeguard for Trojans is to control the entire development chain, this can quickly become impractical. Therefore, we must ensure that we can detect malicious circuits inserted inconspicuously within our FPGA designs. Understanding that our FPGA designs will only increase in complexity, this detection could greatly benefit from applied AI techniques to facilitate detection. Prospectively, AI techniques for controls that apply to other attack techniques also need to be researched.
- *Isolation and segmentation*: Multiple attacks described in Section 5 occurred due to lateral movement or elevation of privilege. A notable example is the use case of FPGA devices in

cloud computing, where power distribution networks are used to perform side-channel and FI attacks and to establish covert channels. Furthermore, when it comes to complex systems such as SoC FPGA devices, their large attack surface gives rise to multiple entry points for attackers, who can then freely access the sensitive designs within the device's configuration memory. To augment security, manufacturers need to privilege practices of isolation and segmentation in their devices.

8 CONCLUSION

This literature review has provided an up-to-date outlook on cybersecurity issues affecting FPGA and SoC FPGA devices and introduced a strategy to help developers effectively apply controls to their systems.

Based primarily on the IDDIL/ATC methodology presented by Muchin and Fitch [114], our strategy sought to define a generic threat model that SoC FPGA developers could adapt for specific architectures and operational environments. Having studied the architectures of the AMD-Xilinx Zynq-7000 and Intel Stratix 10, we observed multiple similarities between these two competing devices. However, what implementation differences there are can drastically affect how we apply controls. This factor becomes increasingly important when we transition from FPGA to SoC FPGA devices. One cannot overlook the added complexity of the HPS and the level of access it grants.

In tackling potential attack vectors, we meticulously reviewed the work of researchers who have demonstrated both active and passive attacks that can break the confidentiality, integrity, and availability of FPGA and SoC FPGA devices. Although manufacturers provide passive security measures that one can easily apply to their FPGA and SoC FPGA devices, these do not translate into foolproof systems. Physical security and active security measures will play a significant role in protecting the device from malicious actors. Developers must carefully study known cybersecurity issues and validate how they apply to their attack surface. From here, developers should use controls based on the threat posed by identified attack vectors and threat actors.

ACKNOWLEDGMENTS

The authors would like to thank Thales Digital Solutions, Prompt Innov, and Canada's Natural Sciences and Engineering Research Council (NSERC) for supporting this work. They would also like to acknowledge the reviewers who provided thoughtful comments that helped significantly improve this work.

REFERENCES

[1] Karim M. Abdellatif and Olivier Hériveaux. 2020. *SiliconToaster: A Cheap and Programmable EM Injector for Extracting Secrets.* Cryptology ePrint Archive, Report 2020/1115. https://eprint.iacr.org/2020/1115.

[2] Michel Agoyan, Jean-Max Dutertre, David Naccache, Bruno Robisson, and Assia Tria. 2010. When clocks fail: On critical paths and clock faults. In *Smart Card Research and Advanced Application*, Dieter Gollmann, Jean-Louis Lanet, and Julien Iguchi-Cartigny (Eds.). Springer, Berlin, Germany, 182–193.

[3] Qazi Arbab Ahmed, Tobias Wiersema, and Marco Platzner. 2021. Malicious routing: Circumventing bitstream-level verification for FPGAs. In *Proceedings of the 2021 Design, Automation, and Test in Europe Conference and Exhibition (DATE'21).* 1490–1495. https://doi.org/10.23919/DATE51398.2021.9474026

[4] Mahbub Alam, Shahin Tajik, Fatemeh Ganji, Mark Tehranipoor, and Domenic Forte. 2019. RAM-Jam: Remote temperature and voltage fault attack on FPGAs using memory collisions. In *Proceedings of the 2019 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC'19).* 48–55. https://doi.org/10.1109/FDTC.2019.00015

[5] Amazonaws.com. 2020. PolarFire FPGA Design Flow Libero SoC (v12.6). Retrieved January 30, 2022 from http://coredocs.s3.amazonaws.com/Libero/12_6_0/Tool/pf_des_flow_ug.pdf.

[6] AMD-Xilinx. 2007. UG002—Virtex-II Platform FPGA User Guide. Retrieved January 30, 2022 from https://www.xilinx.com/support/documentation/user_guides/ug002.pdf.

[7] AMD-Xilinx. 2014. UG109—Programming ARM TrustZone Architecture on the Xilinx Zynq-7000 All Programmable SoC (v1.0). Retrieved September 23, 2022 from https://docs.xilinx.com/v/u/en-US/ug1019-zynq-trustzone.
[8] AMD-Xilinx. 2019. UG585 Zynq-7000 SoC Technical Reference Manual v1.12.2. Retrieved January 30, 2022 from https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf.
[9] AMD-Xilinx. 2020. UG892—Vivado Design Flows Overview. Retrieved January 30, 2022 from https://www.xilinx.com/support/documentation/sw_manuals/xilinx2020_2/ug892-vivado-design-flows-overview.pdf.
[10] AMD-Xilinx. 2022. Design Advisory for Zynq-7000: FSBL Authentication Attack. Retrieved September 23, 2022 from https://support.xilinx.com/s/article/76974?language=en_US.
[11] N. Nalla Anandakumar, Mohammad S. Hashmi, and Mark Tehranipoor. 2021. FPGA-based physical unclonable functions: A comprehensive overview of theory and architectures. *Integration* 81 (2021), 175–194. https://doi.org/10.1016/j.vlsi.2021.06.001
[12] Md. Armanuzzaman and Ziming Zhao. 2022. BYOTee: Towards building your own trusted execution environments using FPGA. *arXiv: 2203.04214* (2022). https://doi.org/10.48550/ARXIV.2203.04214
[13] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan. 2006. The Sorcerer's apprentice guide to fault attacks. *Proceedings of the IEEE* 94, 2 (2006), 370–382.
[14] Elaine Barker. 2020. Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms. Retrieved January 30, 2022 from http://dx.doi.org/10.6028/NIST.SP.800-175B
[15] Davide B. Bartolini, Philipp Miedl, and Lothar Thiele. 2016. On the capacity of thermal covert channels in multicores. In *Proceedings of the 11th European Conference on Computer Systems (EuroSys'16)*. ACM, New York, NY, Article 24, 16 pages. https://doi.org/10.1145/2901318.2901322
[16] Pierre Bayon, Lilian Bossuet, Alain Aubert, and Viktor Fischer. 2013. Electromagnetic analysis on ring oscillator-based true random number generators. In *Proceedings of the 2013 IEEE International Symposium on Circuits and Systems (ISCAS'13)*. 1954–1957. https://doi.org/10.1109/ISCAS.2013.6572251
[17] Pierre Bayon, Lilian Bossuet, Alain Aubert, Viktor Fischer, François Poucheret, Bruno Robisson, and Philippe Maurine. 2012. Contactless electromagnetic active attack on ring oscillator based true random number generator. In *Constructive Side-Channel Analysis and Secure Design*, Werner Schindler and Sorin A. Huss (Eds.). Springer, Berlin, Germany, 151–166.
[18] E. M. Benhani, L. Bossuet, and A. Aubert. 2019. The security of ARM TrustZone in a FPGA-based SoC. *IEEE Transactions on Computers* 68, 8 (Aug. 2019), 1238–1248. https://doi.org/10.1109/TC.2019.2900235
[19] El Mehdi Benhani, Cedric Marchand, Alain Aubert, and Lilian Bossuet. 2017. On the security evaluation of the ARM TrustZone extension in a heterogeneous SoC. In *Proceedings of the 2017 30th IEEE International System-on-Chip Conference (SOCC'17)*. 108–113. https://doi.org/10.1109/SOCC.2017.8226018
[20] F. Benz, A. Seffrin, and S. A. Huss. 2012. Bil: A tool-chain for bitstream reverse-engineering. In *Proceedings of the 22nd International Conference on Field Programmable Logic and Applications (FPL'12)*. 735–738.
[21] D. Black. 2019. Xilinx Says Its New FPGA Is World's Largest. Retrieved January 30, 2022 from https://www.enterpriseai.news/2019/08/21/xilinx-says-its-new-fpga-is-worlds-largest/.
[22] Jakub Breier, Wei He, Dirmanto Jap, Shivam Bhasin, and Anupam Chattopadhyay. 2017. Attacks in reality: The limits of concurrent error detection codes against laser fault injection. *Journal of Hardware and Systems Security* 1, 4 (Dec. 2017), 298–310. https://doi.org/10.1007/s41635-017-0020-3
[23] Eric Brier, Christophe Clavier, and Francis Olivier. 2004. Correlation power analysis with a leakage model. In *Cryptographic Hardware and Embedded Systems—CHES 2004*, Marc Joye and Jean-Jacques Quisquater (Eds.). Springer, Berlin, Germany, 16–29.
[24] G. Canivet, P. Maistri, R. Leveugle, J. Clédière, F. Valette, and M. Renaudin. 2011. Glitch and laser fault attacks onto a secure AES implementation on a SRAM-based FPGA. *Journal of Cryptology* 24, 2 (April 2011), 247–268. https://doi.org/10.1007/s00145-010-9083-9
[25] Rajat Subhra Chakraborty, Indrasish Saha, Ayan Palchaudhuri, and Gowtham Kumar Naik. 2013. Hardware Trojan insertion by direct modification of FPGA configuration bitstream. *IEEE Design & Test* 30, 2 (2013), 45–54. https://doi.org/10.1109/MDT.2013.2247460
[26] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. 2002. Template attacks. In *Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems (CHES'02)*.
[27] Minzhen Chen and Peng Liu. 2021. A deep learning-based FPGA function block detection method with bitstream to image transformation. *IEEE Access* 9 (2021), 99794–99804. https://doi.org/10.1109/ACCESS.2021.3096664
[28] Chithra Chithra, J. Kokila, and N. Ramasubramanian. 2020. Detection of hardware Trojans using machine learning in SoC FPGAs. In *Proceedings of the 2020 IEEE International Conference on Electronics, Computing, and Communication Technologies (CONECCT'20)*. 1–7. https://doi.org/10.1109/CONECCT50063.2020.9198475
[29] Kwen-Siong Chong, Jun-Sheng Ng, Juncheng Chen, Ne Kyaw Zwa Lwin, Nay Aung Kyaw, Weng-Geng Ho, Joseph Chang, and Bah-Hwee Gwee. 2021. Dual-hiding side-channel-attack resistant FPGA-based asynchronous-logic AES:

Design, countermeasures and evaluation. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 11, 2 (2021), 343–356. https://doi.org/10.1109/JETCAS.2021.3077887

[30] Clifford. 2021. Project IceStorm. Retrieved February 9, 2021 from http://www.clifford.at/icestorm.

[31] Ang Cui and Rick Housley. 2017. BADFET: Defeating modern secure boot using second-order pulsed electromagnetic fault injection. In *Proceedings of the 11th USENIX Workshop on Offensive Technologies (WOOT'17)*. https://www.usenix.org/conference/woot17/workshop-program/presentation/cui.

[32] Wafi Danesh, Joshua Banago, and Mostafizur Rahman. 2021. Turning the table: Using bitstream reverse engineering to detect FPGA Trojans. *Journal of Hardware and Systems Security* 5, 3 (Dec. 2021), 237–246. https://doi.org/10.1007/s41635-021-00122-4

[33] Amine Dehbaoui, Jean-Max Dutertre, Bruno Robisson, and Assia Tria. 2012. Electromagnetic transient faults injection on a hardware and a software implementations of AES. In *Proceedings of the 2012 Workshop on Fault Diagnosis and Tolerance in Cryptography*. 7–15. https://doi.org/10.1109/FDTC.2012.15

[34] Ghada Dessouky, Ahmad-Reza Sadeghi, and Shaza Zeitouni. 2021. SoK: Secure FPGA multi-tenancy in the cloud: Challenges and opportunities. In *Proceedings of the 2021 IEEE European Symposium on Security and Privacy (EuroS&P'21)*. 487–506. https://doi.org/10.1109/EuroSP51992.2021.00040

[35] Marion Doulcier-Verdier, Jean-Max Dutertre, Jacques Fournier, Jean-Baptiste Rigaud, Bruno Robisson, and Assia Tria. 2011. A side-channel and fault-attack resistant AES circuit working on duplicated complemented values. In *Proceedings of the 2011 IEEE International Solid-State Circuits Conference*. 274–276. https://doi.org/10.1109/ISSCC.2011.5746316

[36] R. Druyer, L. Torres, P. Benoit, P. V. Bonzom, and P. Le-Quere. 2015. A survey on security features in modern FPGAs. In *Proceedings of the 2015 10th International Symposium on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC'15)*. IEEE, Los Alamitos, CA, 1–8. http://ieeexplore.ieee.org/document/7238102/.

[37] Shijin Duan, Wenhao Wang, Yukui Luo, and Xiaolin Xu. 2021. A survey of recent attacks and mitigation on FPGA systems. In *Proceedings of the 2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI'21)*. 284–289. https://doi.org/10.1109/ISVLSI51109.2021.00059

[38] Anuj Dubey, Rosario Cammarota, and Aydin Aysu. 2020. MaskedNet: The first hardware inference engine aiming power side-channel protection. In *Proceedings of the 2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST'20)*. 197–208. https://doi.org/10.1109/HOST45689.2020.9300276

[39] Adam Duncan, Fahim Rahman, Andrew Lukefahr, Farimah Farahmandi, and Mark Tehranipoor. 2019. FPGA bitstream security: A day in the life. In *Proceedings of the 2019 IEEE International Test Conference (ITC'19)*. 1–10. https://doi.org/10.1109/ITC44170.2019.9000145

[40] Matthias Dyer, Christian Plessl, and Marco Platzner. 2002. Partially reconfigurable cores for Xilinx Virtex. In *Field-Programmable Logic and Applications: Reconfigurable Computing Is Going Mainstream*. Lecture Notes in Computer Science, Vol. 2438. Springer, 292–301. http://link.springer.com/10.1007/3-540-46117-5_31.

[41] Rana Elnaggar, Sayak Ray, Majid Sabbagh, Bilgiday Yuce, Terry Wang, and Jason Fung. 2021. OPAL: On-the-go physical attack lab to evaluate power side-channel vulnerabilities on FPGAs. In *Proceedings of the 2021 IEEE Physical Assurance and Inspection of Electronics (PAINE'21)*. 1–8. https://doi.org/10.1109/PAINE54418.2021.9707701

[42] Maik Ender, Amir Moradi, and Christof Paar. 2020. The unpatchable silicon: A full break of the bitstream encryption of Xilinx 7-series FPGAs. In *Proceedings of the 29th USENIX Security Symposium (USENIX Security'20)*.

[43] FIRST. 2015. CVSS v3.1 Specification Document—Revision 1. Retrieved September 23, 2022 from https://www.first.org/cvss/v3-1/cvss-v31-specification_r1.pdf.

[44] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. 2008. Mutual information analysis. In *Cryptographic Hardware and Embedded Systems—CHES 2008*, Elisabeth Oswald and Pankaj Rohatgi (Eds.). Springer, Berlin, Germany, 426–442.

[45] Ognjen Glamočanin, Louis Coulon, Francesco Regazzoni, and Mirjana Stojilović. 2020. Built-in self-evaluation of first-order power side-channel leakage for FPGAs. In *Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA'20)*. ACM, New York, NY, 204–210. https://doi.org/10.1145/3373087.3375318

[46] Dennis R. E. Gnad, Cong Dang Khoa Nguyen, Syed Hashim Gillani, and Mehdi B. Tahoori. 2019. *Voltage-Based Covert Channels Using FPGAs*. Cryptology ePrint Archive, Report 2019/1394. https://ia.cr/2019/1394.

[47] Dennis R. E. Gnad, Fabian Oboril, and Mehdi B. Tahoori. 2017. Voltage drop-based fault attacks on FPGAs using valid bitstreams. In *Proceedings of the 2017 27th International Conference on Field Programmable Logic and Applications (FPL'17)*. 1–7.

[48] Joseph Gravellier, Jean-Max Dutertre, Yannick Teglia, Philippe Loubet Moundi, and Francis Olivier. 2020. Remote side-channel attacks on heterogeneous SoC. In *Smart Card Research and Advanced Applications*, Sonia Belaïd and Tim Güneysu (Eds.). Springer International Publishing, Cham, Switzerland, 109–125.

[49] Mathieu Gross, Nisha Jacob, Andreas Zankl, and Georg Sigl. 2022. Breaking TrustZone memory isolation and secure boot through malicious hardware on a modern FPGA-SoC. *Journal of Cryptographic Engineering* 12, 2 (June 2022), 181–196. https://doi.org/10.1007/s13389-021-00273-8

[50] Y. Hamadi and D. Merceron. 1997. Reconfigurable architectures: A new vision for optimization problems. In *Principles and Practice of Constraint Programming-CP97*. Lecture Notes in Computer Science, Vol. 1330. Springer, 209–221.
[51] Wei He, Jakub Breier, and Shivam Bhasin. 2016. Cheap and cheerful: A low-cost digital sensor for detecting laser fault injection attacks. In *Security, Privacy, and Applied Cryptography Engineering*, Claude Carlet, M. Anwar Hasan, and Vishal Saraswat (Eds.). Springer International Publishing, Cham, Switzerland, 27–46.
[52] Wei He, Jakub Breier, Shivam Bhasin, Noriyuki Miura, and Makoto Nagata. 2016. Ring oscillator under laser: Potential of PLL-based countermeasure against laser fault injection. In *Proceedings of the 2016 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC'16)*. 102–113. https://doi.org/10.1109/FDTC.2016.13
[53] Wei He, Jakub Breier, Shivam Bhasin, Noriyuki Miura, and Makoto Nagata. 2017. An FPGA-compatible PLL-based sensor against fault injection attack. In *Proceedings of the 2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC'17)*. 39–40. https://doi.org/10.1109/ASPDAC.2017.7858291
[54] Tamzidul Hoque, Kai Yang, Robert Karam, Shahin Tajik, Domenic Forte, Mark Tehranipoor, and Swarup Bhunia. 2019. Hidden in plaintext: An obfuscation-based countermeasure against FPGA bitstream tampering attacks. *ACM Transactions on Design Automation of Electronic Systems* 25, 1 (Nov. 2019), Article 4, 32 pages. https://doi.org/10.1145/3361147
[55] Karen Horovitz and Ryan Kenny. 2018. Intel FPGA Secure Device Manager. Retrieved January 30, 2022 from https://apps.dtic.mil/sti/pdfs/AD1052301.pdf.
[56] Zhao Huang, Quan Wang, Yin Chen, and Xiaohong Jiang. 2020. A survey on machine learning against hardware Trojan attacks: Recent advances and challenges. *IEEE Access* 8 (2020), 10796–10826. https://doi.org/10.1109/ACCESS.2020.2965016
[57] T. Iakymchuk, Maciej Nikodem, and Krzysztof Kepa. 2011. Temperature-based covert channel in FPGA systems. In *Proceedings of the 6th International Workshop on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC'11)*. 1–7.
[58] IEEE. 2013. IEEE standard for test access port and boundary-scan architecture. *IEEE Std 1149.1-2013 (Revision of IEEE Std 1149.1-2001)* (2013), 1–444.
[59] Intel. 2020. Intel® Stratix® 10 FPGA for Intel® Quartus® Prime Pro Advisory. Retrieved January 30, 2022 from https://www.intel.com/content/www/us/en/security-center/advisory/intel-sa-00388.html.
[60] Intel. 2020. UG20132—Intel Quartus Prime Pro Edition Design Compilation. Retrieved January 30, 2022 from https://www.intel.la/content/dam/www/programmable/us/en/pdfs/literature/ug/ug-qpp-compiler.pdf.
[61] Intel. 2021. UG20101—Intel Stratix 10 SoC FPGA Boot User Guide (v21.1). Retrieved June 22, 2021 from https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug-s10-soc-boot.pdf.
[62] Intel. 2021. UGS10CONFIG—Intel Stratix 10 Configuration User Guide (v21.1). Retrieved June 22, 2021 from https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/stratix-10/ug-s10-config.pdf.
[63] Vishnuvardhan V. Iyer and Ali E. Yilmaz. 2019. An adaptive acquisition approach to localize electromagnetic information leakage from cryptographic modules. In *Proceedings of the 2019 IEEE Texas Symposium on Wireless and Microwave Circuits and Systems (WMCS'19)*. 1–6. https://doi.org/10.1109/WMCaS.2019.8732510
[64] Nisha Jacob, Johann Heyszl, Andreas Zankl, Carsten Rolfes, and Georg Sigl. 2017. How to break secure boot on FPGA SoCs through malicious hardware. In *Cryptographic Hardware and Embedded Systems—CHES 2017*. Lecture Notes in Computer Science, Vol. 10529. Springer, 425–442.
[65] Minyoung Jeong, Jaeheum Lee, Eungu Jung, Young Hwan Kim, and Kyoungrok Cho. 2018. Extract LUT logics from a downloaded bitstream data in FPGA. In *Proceedings of the 2018 IEEE International Symposium on Circuits and Systems (ISCAS'18)*. 1–5. https://doi.org/10.1109/ISCAS.2018.8350950
[66] Chenglu Jin, Vasudev Gohil, Ramesh Karri, and Jeyavijayan Rajendran. 2020. Security of cloud FPGAs: A survey. *arXiv:2005.04867* (2020). https://doi.org/10.48550/ARXIV.2005.04867
[67] Jens-Peter Kaps, Kris Gaj, Abubakr Abdulgadir, and Kamyar Mohajerani. 2022. *General Framework for Evaluating LWC Finalists in Terms of Resistance to Side-Channel Attacks*. National Institute of Standards and Technology.
[68] Duško Karaklajić, Jörn-Marc Schmidt, and Ingrid Verbauwhede. 2013. Hardware designer's guide to fault attacks. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 21, 12 (2013), 2295–2306.
[69] Toshihiro Katashita, Yohei Hori, Hirofumi Sakane, and Akashi Satoh. 2011. *Side-Channel Attack Standard Evaluation Board SASEBO-W for Smartcard Testing*. National Institute of Standards and Technology.
[70] Toshihiro Katashita, Akihiko Sasaki, and Yohei Hori. 2013. A novel smart card development platform for evaluating physical attacks and PUFs. In *Proceedings of the 2013 IEEE 2nd Global Conference on Consumer Electronics (GCCE'13)*. 37–39. https://doi.org/10.1109/GCCE.2013.6664860
[71] Yongseen Kim, Eun-Gu Jung, and ChangKyun Kim. 2021. Bitstream reverse engineering of Microsemi's VersaTile-based FPGAs. In *Proceedings of the 2021 IEEE Physical Assurance and Inspection of Electronics (PAINE'21)*. 1–8. https://doi.org/10.1109/PAINE54418.2021.9707700

[72] Paul Kocher, Joshua Jaffe, Benjamin Jun, and Pankaj Rohatgi. 2011. Introduction to differential power analysis. *Journal of Cryptographic Engineering* 1, 1 (April 2011), 5–27. https://doi.org/10.1007/s13389-011-0006-y

[73] Paul C. Kocher. 1996. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in Cryptology—CRYPTO'96*, Neal Koblitz (Ed.). Springer, Berlin, Germany, 104–113.

[74] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Differential power analysis. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO'99)*. 388–397.

[75] Jakub Korczyc and Andrzej Krasniewski. 2012. Evaluation of susceptibility of FPGA-based circuits to fault injection attacks based on clock glitching. In *Proceedings of the 2012 IEEE 15th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS'12)*. 171–174. https://doi.org/10.1109/DDECS.2012.6219047

[76] Thilo Krachenfels, Heiko Lohrke, Jean-Pierre Seifert, Enrico Dietz, Sven Frohmann, and Heinz-Wilhelm Hübers. 2019. Evaluation of low-cost thermal laser stimulation for data extraction and key readout. *Journal of Hardware and Systems Security* 4, 1 (Nov. 2019), 24–33. https://doi.org/10.1007/s41635-019-00083-9

[77] Jonas Krautter, Dennis R. E. Gnad, Falk Schellenberg, Amir Moradi, and Mehdi B. Tahoori. 2019. Active fences against voltage-based side channels in multi-tenant FPGAs. In *Proceedings of the 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD'19)*. 1–8. https://doi.org/10.1109/ICCAD45719.2019.8942094

[78] Jonas Krautter, Dennis R. E. Gnad, and Mehdi B. Tahoori. 2018. FPGAhammer: Remote voltage fault attacks on shared FPGAs, suitable for DFA on AES. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2018, 3 (8 2018), 44–68. https://tches.iacr.org/index.php/TCHES/article/view/7268.

[79] Lattice. 2020. Lattice Diamond 3.12 User Guide. Retrieved January 30, 2022 from http://www.latticesemi.com/view_document?document_id=53077.

[80] Lattice. 2020. TN02001—iCE40 Programming and Configuration (v3.2). Retrieved January 30, 2022 from http://www.latticesemi.com/view_document?document_id=46502.

[81] Adrien Le Masle, Gary C. T. Chow, and Wayne Luk. 2011. Constant power reconfigurable computing. In *Proceedings of the 2011 International Conference on Field-Programmable Technology*. 1–8. https://doi.org/10.1109/FPT.2011.6132682

[82] P. H. W. Leong, C. W. Sham, W. C. Wong, H. Y. Wong, W. S. Yuen, and M. P. Leong. 2001. A bitstream reconfigurable FPGA implementation of the WSAT algorithm. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 9, 1 (Feb. 2001), 197–201.

[83] Wenye Liu, Chip-Hong Chang, Fan Zhang, and Xiaoxuan Lou. 2020. Imperceptible misclassification attack on deep learning accelerator by glitch injection. In *Proceedings of the 2020 57th ACM/IEEE Design Automation Conference (DAC'20)*. 1–6. https://doi.org/10.1109/DAC18072.2020.9218577

[84] Heiko Lohrke, Shahin Tajik, Christian Boit, and Jean-Pierre Seifert. 2016. No place to hide: Contactless probing of secret data on FPGAs. In *Cryptographic Hardware and Embedded Systems—CHES 2016*, Benedikt Gierlichs and Axel Y. Poschmann (Eds.). Springer, Berlin, Germany, 147–167.

[85] Heiko Lohrke, Shahin Tajik, Thilo Krachenfels, Christian Boit, and Jean-Pierre Seifert. 2018. Key extraction using thermal laser stimulation: A case study on Xilinx ultrascale FPGAs. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2018, 3 (Aug. 2018), 573–595. https://doi.org/10.13154/tches.v2018.i3.573-595

[86] Ting Lu, Ryna Kenny, and Sean Atsatt. 2021. WP01252—Secure Device Manager for Intel® Stratix® 10 Devices Provides FPGA and SoC Security. Retrieved January 30, 2022 from https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/wp/wp-01252-secure-device-manager-for-fpga-soc-security.pdf.

[87] Yukui Luo, Shijin Duan, and Xiaolin Xu. 2021. FPGAPRO: A defense framework against crosstalk-induced secret leakage in FPGA. *ACM Transactions on Design Automation of Electronic Systems* 27, 3 (Nov. 2021), Article 24, 31 pages. https://doi.org/10.1145/3491214

[88] Yukui Luo, Cheng Gongye, Yunsi Fei, and Xiaolin Xu. 2021. DeepStrike: Remotely-guided fault injection attacks on DNN accelerator in cloud-FPGA. In *Proceedings of the 2021 58th ACM/IEEE Design Automation Conference (DAC'21)*. 295–300. https://doi.org/10.1109/DAC18074.2021.9586262

[89] Yukui Luo and Xiaolin Xu. 2019. HILL: A hardware isolation framework against information leakage on multi-tenant FPGA long-wires. In *Proceedings of the 2019 International Conference on Field-Programmable Technology (ICFPT'19)*. 331–334. https://doi.org/10.1109/ICFPT47387.2019.00060

[90] Maxime Madau, Michel Agoyan, Josep Balasch, Miloš Grujić, Patrick Haddad, Philippe Maurine, Vladimir Rožić, Dave Singelée, Bohan Yang, and Ingrid Verbauwhede. 2018. The impact of pulsed electromagnetic fault injection on true random number generators. In *Proceedings of the 2018 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC'18)*. 43–48. https://doi.org/10.1109/FDTC.2018.00015

[91] Dina Mahmoud and Mirjana Stojilović. 2019. Timing violation induced faults in multi-tenant FPGAs. In *Proceedings of the 2019 Design, Automation, and Test in Europe Conference and Exhibition (DATE'19)*. 1745–1750. https://doi.org/10.23919/DATE.2019.8715263

[92] Dina G. Mahmoud, Vincent Lenders, and Mirjana Stojilović. 2022. Electrical-level attacks on CPUs, FPGAs, and GPUs: Survey and implications in the heterogeneous era. *ACM Computing Surveys* 55, 3 (Feb. 2022), Article 58, 40 pages. https://doi.org/10.1145/3498337

[93] Paolo Maistri and Régis Leveugle. 2008. Double-data-rate computation as a countermeasure against fault analysis. *IEEE Transactions on Computers* 57, 11 (2008), 1528–1539. https://doi.org/10.1109/TC.2008.149

[94] Macarena C. Martínez-Rodríguez, Ignacio M. Delgado-Lozano, and Billy Bob Brumley. 2021. SoK: Remote power analysis. In *Proceedings of the 16th International Conference on Availability, Reliability, and Security (ARES'21)*. ACM, New York, NY, Article 7, 12 pages. https://doi.org/10.1145/3465481.3465773

[95] Honorio Martín, Thomas Korak, Enrique San Millán, and Michael Hutter. 2015. Fault attacks on STRNGs: Impact of glitches, temperature, and underpowering on randomness. *IEEE Transactions on Information Forensics and Security* 10, 2 (2015), 266–277. https://doi.org/10.1109/TIFS.2014.2374072

[96] Ramya Jayaram Masti, Devendra Rai, Aanjhan Ranganathan, Christian Müller, Lothar Thiele, and Srdjan Capkun. 2015. Thermal covert channels on multi-core platforms. In *Proceedings of the 24th USENIX Security Symposium (USENIX Security'15)*. https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/masti.

[97] Kaspar Matas, Tuan La, Nikola Grunchevski, Khoa Pham, and Dirk Koch. 2020. Invited tutorial: FPGA hardware security for datacenters and beyond. In *Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA'20)*. ACM, New York, NY, 11–20. https://doi.org/10.1145/3373087.3375390

[98] Masato Matsubayashi, Akashi Satoh, and Jun Ishii. 2016. Clock glitch generator on SAKURA-G for fault injection attack against a cryptographic circuit. In *Proceedings of the 2016 IEEE 5th Global Conference on Consumer Electronics*. 1–4. https://doi.org/10.1109/GCCE.2016.7800490

[99] Nele Mentens, Benedikt Gierlichs, and Ingrid Verbauwhede. 2008. Power and fault analysis resistance in hardware through dynamic reconfiguration. In *Cryptographic Hardware and Embedded Systems—CHES 2008*, Elisabeth Oswald and Pankaj Rohatgi (Eds.). Springer, Berlin, Germany, 346–362.

[100] Microsemi. 2019. Microchip—FPGA and SoC Product Catalog. Retrieved January 30, 2022 from https://www.microsemi.com/document-portal/doc_download/1244242-fpga-soc-catalog.

[101] Seyedeh Sharareh Mirzargar, Gaiëtan Renault, Andrea Guerrieri, and Mirjana Stojilović. 2020. Nonintrusive and adaptive monitoring for locating voltage attacks in virtualized FPGAs. In *Proceedings of the 2020 International Conference on Field-Programmable Technology (ICFPT'20)*. 288–289. https://doi.org/10.1109/ICFPT51103.2020.00050

[102] Seyedeh Sharareh Mirzargar and Mirjana Stojilović. 2019. Physical side-channel attacks and covert communication on FPGAs: A survey. In *Proceedings of the 2019 29th International Conference on Field Programmable Logic and Applications (FPL'19)*. 202–210. https://doi.org/10.1109/FPL.2019.00039

[103] MITRE. 2020. CVE-2020-12312. Available from MITRE, CVE-ID CVE-2020-12312. Retrieved May 21, 2022 from https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-12312.

[104] MITRE. 2020. CVE-2020-8737. Available from MITRE, CVE-ID CVE-2020-8737. Retrieved May 21, 2022 from https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-8737.

[105] MITRE. 2021. CVE-2021-27208. Available from MITRE, CVE-ID CVE-2021-27208. Retrieved May 21, 2022 from https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-27208.

[106] MITRE. 2022. CVE. Retrieved May 21, 2022 from https://cve.mitre.org/.

[107] MITRE. 2022. CVE-2021-44850. Available from MITRE, CVE-ID CVE-2021-44850. Retrieved May 21, 2022 from https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-44850.

[108] MITRE. 2022. CVE-2022-23822. Available from MITRE, CVE-ID CVE-2022-23822. Retrieved May 21, 2022 from https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-23822.

[109] Amir Moradi, Alessandro Barenghi, Timo Kasper, and Christof Paar. 2011. On the vulnerability of FPGA bitstream encryption against power analysis attacks: Extracting keys from Xilinx Virtex-II FPGAs. In *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS'11)*. ACM, New York, NY, 111–124. https://doi.org/10.1145/2046707.2046722

[110] Amir Moradi, Markus Kasper, and Christof Paar. 2012. Black-box side-channel attacks highlight the importance of countermeasures—An analysis of the Xilinx Virtex-4 and Virtex-5 bitstream encryption mechanism. In *Topics in Cryptology—CT-RSA 2012*. Lecture Notes in Computer Science, Vol. 7178. Springer, 1–18.

[111] Amir Moradi, David Oswald, Christof Paar, and Pawel Swierczynski. 2013. Side-channel attacks on the bitstream encryption mechanism of Altera Stratix II: Facilitating black-box analysis using software reverse-engineering. In *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA'13)*. ACM, New York, NY, 91–100. https://doi.org/10.1145/2435264.2435282

[112] Amir Moradi and Tobias Schneider. 2016. Improved side-channel analysis attacks on Xilinx bitstream encryption of 5, 6, and 7 series. In *Constructive Side-Channel Analysis and Secure Design*, François-Xavier Standaert and Elisabeth Oswald (Eds.). Springer International Publishing, Cham, Switzerland, 71–87.

[113] Michail Moraitis and Elena Dubrova. 2020. FPGA bitstream modification with interconnect in mind. In *Proceedings of the Conference on Hardware and Architectural Support for Security and Privacy (HASP'20)*. ACM, New York, NY, Article 5, 9 pages. https://doi.org/10.1145/3458903.3458908

[114] Michael Muckin and Scott Fitch. 2019. A Threat-Driven Approach to Cyber Security. Retrieved January 30, 2022 from https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/LM-White-Paper-Threat-Driven-Approach.pdf.

[115] Shyamapada Mukherjee, Swapnanil kr Saikia, Stuti Anand, Ritu Chouhan, and Hiresh Das. 2021. A counter measure to prevent timing-based side-channel attack on FPGA. In *Proceedings of the 2021 6th International Conference on Communication and Electronics Systems (ICCES'21)*. 983–988. https://doi.org/10.1109/ICCES51350.2021.9489054

[116] Kevin E. Murray, Mohamed A. Elgammal, Vaughn Betz, Tim Ansell, Keith Rothman, and Alessandro Comodi. 2020. SymbiFlow and VPR: An open-source design flow for commercial and novel FPGAs. *IEEE Micro* 40, 4 (2020), 49–57.

[117] Kalle Ngo, Elena Dubrova, and Michail Moraitis. 2020. Attacking Trivium at the bitstream level. In *Proceedings of the 2020 IEEE 38th International Conference on Computer Design (ICCD'20)*. 640–647. https://doi.org/10.1109/ICCD50377.2020.00110

[118] Cong Nguyen. 2018. *Voltage-Based Covert Channel Communication between Logically Separated IP Cores in FPGAs*. Bachelor's Thesis. Karlsruhe Institute of Technology. https://cdnc.itec.kit.edu/downloads/Other/online_thesis_final_Khoa_Nguyen.pdf.

[119] NIST. 2001. Announcing the Advanced Encryption Standard. Retrieved January 30, 2022 from https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf.

[120] NIST. 2022. NVD. Retrieved January 30, 2022 from https://nvd.nist.gov/.

[121] Jean-Baptiste Note and Éric Rannaud. 2008. From the bitstream to the netlist. In *Proceedings of the 16th International ACM/SIGDA Symposium on Field Programmable Gate Arrays (FPGA'08)*. ACM, New York, NY, 264. https://doi.org/10.1145/1344671.1344729

[122] Colin O'Flynn. 2016. *Fault Injection Using Crowbars on Embedded Systems*. Cryptology ePrint Archive, Paper 2016/810. https://eprint.iacr.org/2016/810.

[123] Colin O'Flynn and Zhizhang (David) Chen. 2014. ChipWhisperer: An open-source platform for hardware embedded security research. In *Constructive Side-Channel Analysis and Secure Design*, Emmanuel Prouff (Ed.). Springer International Publishing, Cham, Switzerland, 243–260.

[124] S. Ordas, L. Guillaume-Sage, and P. Maurine. 2017. Electromagnetic fault injection: The curse of flip-flops. *Journal of Cryptographic Engineering* 7, 3 (Sept. 2017), 183–197. https://doi.org/10.1007/s13389-016-0128-3

[125] Zhixin Pan and Prabhat Mishra. 2022. A survey on hardware vulnerability analysis using machine learning. *IEEE Access* 10 (2022), 49508–49527. https://doi.org/10.1109/ACCESS.2022.3173287

[126] Michael Paquette, Brian Marquis, Rachel Bainbridge, and Joe Chapman. 2021. Visualizing electromagnetic fault injection with timing sensors. In *Proceedings of the 2021 IEEE Physical Assurance and Inspection of Electronics (PAINE'21)*. 1–8. https://doi.org/10.1109/PAINE54418.2021.9707696

[127] Ed Parson. 2018. XAPP1098—Developing Tamper-Resistant Designs with UltraScale and UltraScale+ FPGAs (v1.3). Retrieved January 30, 2022 from https://www.xilinx.com/support/documentation/application_notes/xapp1098-tamper-resist-designs.pdf.

[128] Ed Peterson. 2015. WP468—Leveraging Asymmetric Authentication to Enhance Security-Critical Applications Using Zynq-7000 All Programmable SoCs (v1.0). Retrieved January 30, 2022 from https://www.xilinx.com/support/documentation/white_papers/wp468_asym-auth-zynq-7000.pdf.

[129] Ed Peterson. 2021. XAPP1175—Secure Boot of Zynq-7000 SoC (v2.2). Retrieved January 30, 2022 from https://www.xilinx.com/support/documentation/application_notes/xapp1175_zynq_secure_boot.pdf.

[130] PrjTrellis. 2021. Welcome to Project Trellis—Project Trellis 0.0-672-gf93243b Documentation. Retrieved February 9, 2021 from https://prjtrellis.readthedocs.io/en/latest.

[131] George Provelengios, Daniel Holcomb, and Russell Tessier. 2019. Characterizing power distribution attacks in multi-user FPGA environments. In *Proceedings of the 2019 29th International Conference on Field Programmable Logic and Applications (FPL'19)*. 194–201. https://doi.org/10.1109/FPL.2019.00038

[132] Yifei Qiao, Zhaojun Lu, Hailong Liu, and Zhenglin Liu. 2017. Clock glitch fault injection attacks on an FPGA AES implementation. *Journal of Electrotechnology, Electrical Engineering and Management* 1, 1 (March 2017). http://clausiuspress.com/article/68.html.

[133] Chethan Ramesh, Shivukumar B. Patil, Siva Nishok Dhanuskodi, George Provelengios, Sebastien Pillement, Daniel Holcomb, and Russell Tessier. 2018. FPGA side channel attacks without physical access. In *Proceedings of the 2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM'18)*. 45–52. https://doi.org/10.1109/FCCM.2018.00016

[134] Keyvan Ramezanpour, Paul Ampadu, and William Diehl. 2020. SCAUL: Power side-channel analysis with unsupervised learning. *IEEE Transactions on Computers* 69, 11 (2020), 1626–1638. https://doi.org/10.1109/TC.2020.3013196

[135] Mark Randolph and William Diehl. 2020. Power side-channel attack analysis: A review of 20 years of study for the layman. *Cryptography* 4, 2 (2020), 15. https://www.mdpi.com/2410-387X/4/2/15.

[136] Wei Ren, Junhao Pan, and Deming Chen. 2021. AccGuard: Secure and trusted computation on remote FPGA accelerators. In *Proceedings of the 2021 IEEE International Symposium on Smart Electronic Systems (iSES'21)*. IEEE, Los Alamitos, CA, 378–383. https://doi.org/10.1109/iSES52644.2021.00093

[137] Galen Schretlen. 'Discovering and exploiting CVE-2021-27208, CVE-2021-44850'. ROPchain, 2022, https://blog.ropcha.in.

[138] Paul D. Rosero-Montalvo. 2021. A survey on security concerns and their actual solutions for using FPGAs in cloud computing. In *Proceedings of the 2021 IEEE Latin American Conference on Computational Intelligence (LA-CCI'21)*. 1–6. https://doi.org/10.1109/LA-CCI48322.2021.9769794

[139] Pascal Sasdrich, Amir Moradi, Oliver Mischke, and Tim Güneysu. 2015. Achieving side-channel protection with dynamic logic reconfiguration on modern FPGAs. In *Proceedings of the 2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST'15)*. 130–136. https://doi.org/10.1109/HST.2015.7140251

[140] Falk Schellenberg, Dennis R. E. Gnad, Amir Moradi, and Mehdi B. Tahoori. 2018. An inside job: Remote power analysis attacks on FPGAs. In *Proceedings of the 2018 Design, Automation, and Test in Europe Conference and Exhibition (DATE'18)*. 1111–1116. https://doi.org/10.23919/DATE.2018.8342177

[141] Zeinab Seifoori, Seyedeh Sharareh Mirzargar, and Mirjana Stojilović. 2020. Closing leaks: Routing against crosstalk side-channel attacks. In *Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA'20)*. ACM, New York, NY, 197–203. https://doi.org/10.1145/3373087.3375319

[142] Bodo Selmke, Stefan Brummer, Johann Heyszl, and Georg Sigl. 2016. Precise laser fault injections into 90 nm and 45 nm SRAM-cells. In *Smart Card Research and Advanced Applications*, Naofumi Homma and Marcel Medwed (Eds.). Springer International Publishing, Cham, Switzerland, 193–205.

[143] Bodo Selmke, Johann Heyszl, and Georg Sigl. 2016. Attack on a DFA protected AES by simultaneous laser fault injections. In *Proceedings of the 2016 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC'16)*. 36–46. https://doi.org/10.1109/FDTC.2016.16

[144] Linda L. Shen, Ibrahim Ahmed, and Vaughn Betz. 2019. Fast voltage transients on FPGAs: Impact and mitigation strategies. In *Proceedings of the 2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM'19)*. 271–279. https://doi.org/10.1109/FCCM.2019.00044

[145] Sergei Skorobogatov. 2005. *Semi-Invasive Attacks—A New Approach to Hardware Security Analysis*. Technical Report UCAM-CL-TR-630. Computer Laboratory, University of Cambridge.

[146] Sergei Skorobogatov. 2017. How microprobing can attack encrypted memory. In *Proceedings of the 2017 Euromicro Conference on Digital System Design (DSD'17)*. 244–251.

[147] SOGIS. 2020. Application of Attack Potential to Smartcards and Similar Devices Version 3.1. Retrieved January 30, 2022 from https://www.sogis.eu/documents/cc/domains/sc/JIL-Application-of-Attack-Potential-to-Smartcards-v3-1.pdf.

[148] Simon Steinegger. 2004. *Reconfigurable Hardware OS Prototype—Part FPGA*. Retrieved January 30, 2022 from https://pub.tik.ee.ethz.ch/students/2003-2004-Wi/DA-2004-05.pdf.

[149] Sandeep Sunkavilli, Zhiming Zhang, and Qiaoyan Yu. 2021. New security threats on FPGAs: From FPGA design tools perspective. In *Proceedings of the 2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI'21)*. 278–283. https://doi.org/10.1109/ISVLSI51109.2021.00058

[150] Pawel Swierczynski, Georg T. Becker, Amir Moradi, and Christof Paar. 2018. Bitstream fault injections (BiFI)—Automated fault attacks against SRAM-based FPGAs. *IEEE Transactions on Computers* 67, 3 (2018), 348–360. https://doi.org/10.1109/TC.2016.2646367

[151] Pawel Swierczynski, Marc Fyrbiak, Philipp Koppe, and Christof Paar. 2015. FPGA Trojans through detecting and weakening of cryptographic primitives. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34, 8 (2015), 1236–1249. https://doi.org/10.1109/TCAD.2015.2399455

[152] F4PGA. 2022. Project X-Ray. Retrieved October 10, 2022 from https://f4pga.readthedocs.io/projects/prjxray/en/latest/db_dev_process/readme.html.

[153] F4PGA. 2022. FPGA Assembly (FASM). Retrieved October 10, 2022 from https://fasm.readthedocs.io/en/latest/.

[154] Oliver Söll, Thomas Korak, Michael Muehlberghuber, and Michael Hutter. 2014. EM-based detection of hardware trojans on FPGAs. In *Proceedings of the 2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST'14)*. 84–87. https://doi.org/10.1109/HST.2014.6855574

[155] Shahin Tajik, Julian Fietkau, Heiko Lohrke, Jean-Pierre Seifert, and Christian Boit. 2017. PUFMon: Security monitoring of FPGAs using physically unclonable functions. In *Proceedings of the 2017 IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS'17)*. 186–191. https://doi.org/10.1109/IOLTS.2017.8046216

[156] Shahin Tajik, Heiko Lohrke, Fatemeh Ganji, Jean-Pierre Seifert, and Christian Boit. 2015. Laser fault attack on physically unclonable functions. In *Proceedings of the 2015 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC'15)*. 85–96. https://doi.org/10.1109/FDTC.2015.19

[157] Shahin Tajik, Heiko Lohrke, Jean-Pierre Seifert, and Christian Boit. 2017. On the power of optical contactless probing: Attacking bitstream encryption of FPGAs. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS'17)*. 1661–1674.

[158] Christopher Tarnovsky. 2008. Security failures in secure devices. In *Black Hat DC*, Vol. 74. Black Hat.

[159] Shanquan Tian and Jakub Szefer. 2019. Temporal thermal covert channels in cloud FPGAs. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA'19)*. ACM, New York, NY, 298–303. https://doi.org/10.1145/3289602.3293920

[160] Niek Timmers and Cristofaro Mune. 2017. Escalating privileges in Linux using voltage fault injection. In *Proceedings of the 2017 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC'17)*. 1–8. https://doi.org/10.1109/FDTC.2017.16

[161] Niek Timmers and Albert Spruyt. 2016. Bypassing Secure Boot Using Fault Injection. Retrieved September 23, 2022 from https://www.riscure.com/publication/bypassing-secure-boot-using-fault-injection.

[162] Steve Trimberger and Steve McNeil. 2017. Security of FPGAs in data centers. In *Proceedings of the 2017 IEEE 2nd International Verification and Security Workshop (IVSW'17)*. 117–122. https://doi.org/10.1109/IVSW.2017.8031556

[163] Furkan Turan and Ingrid Verbauwhede. 2020. Trust in FPGA-accelerated cloud computing. *ACM Computing Surveys* 53, 6 (Dec. 2020), Article 128, 28 pages. https://doi.org/10.1145/3419100

[164] Huanyu Wang and Elena Dubrova. 2020. Tandem deep learning side-channel attack against FPGA implementation of AES. In *Proceedings of the 2020 IEEE International Symposium on Smart Electronic Systems (iSES'20)*. 147–150. https://doi.org/10.1109/iSES50453.2020.00041

[165] Huanyu Wang, Domenic Forte, Mark M. Tehranipoor, and Qihang Shi. 2017. Probing attacks on integrated circuits: Challenges and research opportunities. *IEEE Design & Test* 34, 5 (2017), 63–71.

[166] Lingxiao Wei, Bo Luo, Yu Li, Yannan Liu, and Qiang Xu. 2018. I know what you see: Power side-channel attack on convolutional neural network accelerators. In *Proceedings of the 34th Annual Computer Security Applications Conference (ACSAC'18)*. ACM, New York, NY, 393–406. https://doi.org/10.1145/3274694.3274696

[167] Ke Xia, Yukui Luo, Xiaolin Xu, and Sheng Wei. 2021. SGX-FPGA: Trusted execution environment for CPU-FPGA heterogeneous architecture. In *Proceedings of the 2021 58th ACM/IEEE Design Automation Conference (DAC'21)*. IEEE, Los Alamitos, CA, 301–306. https://doi.org/10.1109/DAC18074.2021.9586207

[168] Peng Yang, Fang Luo, Qingyu Ou, and Dawei Zhou. 2020. Design and analysis of clock fault injection for AES. In *Proceedings of the 2020 International Conference on Computer Communication and Network Security (CCNS'20)*. 87–91. https://doi.org/10.1109/CCNS50731.2020.00027

[169] Jing Ye, Yu Hu, and Xiaowei Li. 2018. Hardware Trojan in FPGA CNN accelerator. In *Proceedings of the 2018 IEEE 27th Asian Test Symposium (ATS'18)*. 68–73. https://doi.org/10.1109/ATS.2018.00024

[170] Honggang Yu, Haocheng Ma, Kaichen Yang, Yiqiang Zhao, and Yier Jin. 2020. DeepEM: Deep neural networks model recovery through EM side-channel information leakage. In *Proceedings of the 2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST'20)*. 209–218. https://doi.org/10.1109/HOST45689.2020.9300274

[171] Yang Yu. 2020. Why deep learning makes it difficult to keep secrets in FPGAs. In *Proceedings of the 2020 Workshop in Dynamic and Novel Advances in Machine Learning and Intelligent Cyber Security (DYNAMICS'20)*.

[172] Yang Yu, Michail Moraitis, and Elena Dubrova. 2021. Can deep learning break a true random number generator? *IEEE Transactions on Circuits and Systems II: Express Briefs* 68, 5 (2021), 1710–1714. https://doi.org/10.1109/TCSII.2021.3066338

[173] Wong Hiu Yung, Yuen Wing Seung, Kin Hong Lee, and Philip Heng Wai Leong. 1999. A runtime reconfigurable implementation of the GSAT algorithm. In *Field Programmable Logic and Applications*. Lecture Notes in Computer Science, Vol. 1673. Springer, 526–531.

[174] Jiliang Zhang and Gang Qu. 2019. Recent attacks and defenses on FPGA-based systems. *ACM Transactions on Reconfigurable Technology and Systems* 12, 3 (Sept. 2019), 1–24. https://dl.acm.org/doi/10.1145/3340557

[175] Tao Zhang, Jian Wang, Shize Guo, and Zhe Chen. 2019. A comprehensive FPGA reverse engineering tool-chain: From bitstream to RTL code. *IEEE Access* 7 (2019), 38379–38389. https://ieeexplore.ieee.org/document/8653869/.

[176] Zhiming Zhang, Laurent Njilla, Charles A. Kamhoua, and Qiaoyan Yu. 2019. Thwarting security threats from malicious FPGA tools with novel FPGA-oriented moving target defense. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 27, 3 (2019), 665–678. https://doi.org/10.1109/TVLSI.2018.2879878

[177] Mark Zhao and G. Edward Suh. 2018. FPGA-based remote power side-channel attacks. In *Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP'18)*. 229–244. https://doi.org/10.1109/SP.2018.00049

[178] Pu Zhao, Siyue Wang, Cheng Gongye, Yanzhi Wang, Yunsi Fei, and Xue Lin. 2019. Fault sneaking attack: A stealthy framework for misleading deep neural networks. In *Proceedings of the 56th Annual Design Automation Conference (DAC'19)*. ACM, New York, NY, Article 165, 6 pages. https://doi.org/10.1145/3316781.3317825

[179] Daniel Ziener, Stefan Assmus, and Juurgen Teich. 2006. Identifying FPGA IP-cores based on lookup table content analysis. In *Proceedings of the 2006 International Conference on Field Programmable Logic and Applications*. 1–6.

[180] Loic Zussa, Amine Dehbaoui, Karim Tobich, Jean-Max Dutertre, Philippe Maurine, Ludovic Guillaume-Sage, Jessy Clediere, and Assia Tria. 2014. Efficiency of a glitch detector against electromagnetic fault injection. In *Proceedings of the 2014 Design, Automation, and Test in Europe Conference and Exhibition (DATE'14)*. 1–6. https://doi.org/10.7873/DATE.2014.216

[181] Loïc Zussa, Jean-Max Dutertre, Jessy Clédière, Bruno Robisson, and Assia Tria. 2012. Investigation of timing constraints violation as a fault injection means. In *Proceedings of the 27th Conference on Design of Circuits and Integrated Systems (DCIS'12)*. https://hal-emse.ccsd.cnrs.fr/emse-00742652.

[182] Loïc Zussa, Jean-Max Dutertre, Jessy Clédière, and Assia Tria. 2013. Power supply glitch induced faults on FPGA: An in-depth analysis of the injection mechanism. In *Proceedings of the 2013 IEEE 19th International On-Line Testing Symposium (IOLTS'13)*. 110–115. https://doi.org/10.1109/IOLTS.2013.6604060

Received 13 February 2022; revised 26 July 2022; accepted 14 August 2022