Trust in FPGA Accelerated Cloud Computing

FURKAN TURAN and INGRID VERBAUWHEDE, imec-COSIC, KU Leuven, Belgium

Platforms combining CPUs with FPGAs have become popular as they promise high performance with energy efficiency. This is the result of the combination of FPGA accelerators tuned to the application, with the CPU providing the programming flexibility. Unfortunately, the security of these new platforms has received little attention: the classic software security assumption that hardware is immutable no longer holds. It is expected that attack surfaces will expand and threats will evolve, hence the trust models, and security solutions should be prepared. The attacker model should be enhanced and consider the following three basic entities as the source of threats: applications run by users, accelerators designed by third-party developers and the cloud service providers enabling the computation on their platforms. In our work, we review current trust models and existing security assumptions, and point out their shortcomings. We survey existing research that target secure remote FPGA configuration, the protection of intellectual property, and secure shared use of FPGAs. When combined, these are the foundations to build a solution for secure use of FPGAs in the cloud. In addition to analysing the existing research, we provide discussions on how to improve it, and disclose various concerns that have not been addressed yet.

CCS Concepts: • **Computer systems organization → Reconfigurable computing**; • **Security and privacy → Domain-specific security and privacy architectures**.

Additional Key Words and Phrases: CPU, FPGA, Trust, Security

ACM Reference Format:
Furkan Turan and Ingrid Verbauwhede. 2020. Trust in FPGA Accelerated Cloud Computing. *ACM Comput. Surv.* 1, 1, Article 1 (March 2020), 28 pages.

1 INTRODUCTION

The novel FPGA accelerated computation platforms offer spectacular speed-ups for computation intensive applications because the platform can be tuned to the applications with domain specific acceleration in the FPGA fabric. These platforms are receiving a lot of attention from industry giants (e.g., Intel, Microsoft, Amazon, IBM, Huawei, ...) providing either the platforms and/or FPGA-accelerated web services. However, their solutions are not optimal, lacking a decent trust model and paving the way for security issues. This paper highlights the problems of existing trust models and security features of current FPGAs, surveys the existing research that aims at solving them, and discusses how to improve these solutions specifically for FPGA accelerated cloud.

Traditional computer architectures have the capability to offer isolated execution. It enables various applications and users to share a single platform temporary, spatially and even virtually. Isolation is the key to run code from multiple sources, especially if it is questionable whether they are trusted or not. However, the tightly coupled CPU and FPGA platforms are not yet equipped

Authors' address: Furkan Turan; Ingrid Verbauwhede, imec-COSIC, KU Leuven, Kasteelpark Arenberg 10, bus 2452, Leuven, 3001, Belgium, firstname.lastname@esat.kuleuven.be.

0360-0300/2020/3-ART1 $15.00
https://doi.org/

with similar primitives. Moreover, they lack a mechanism to warrant only trusted designs on the hardware. Consequently, today's platforms restrict users from enjoying the shared use.

The increasing interest in hardware development is exciting, but also a bit concerning as the security lags behind the developed systems. For example, the growing open-source hardware community enables building a platform with license-free modules, but it also makes trusting hardware intractable, since current defence mechanisms do not offer protection from hardware level attacks. Therefore, research for hardware trust and security has become unavoidable. Otherwise, a vulnerability may easily become the weakest point of a platform, and its existence may even lead to declare the whole system as untrusted or insecure.

Examples of achievements with FPGA acceleration appear every day. For example, an online streaming platform Twitch announced [10] 30 fold speed up to deliver 120 frames per-second live over CPU implementation, achieved on Amazon's FPGA accelerated cloud platform. In a more recent example, an FPGA accelerated homomorphic computation (i.e. encrypted data processing for privacy-preserving cloud computing) has been demonstrated [54] to be 5 times faster compared to a GPU implementation on Amazon's Cloud. As a result, there is an incentive to prefer FPGA accelerated cloud for domain specific applications. Hence, more FPGA accelerators are expected to become available, either downloadable from public domain or offered for use on the FPGA accelerated cloud platforms. This increase gives rise to many new security questions. For example, an application will be vulnerable to several attacks, when it searches for available accelerators on a remote platform, finds the one it needs, and requests using that accelerator. Since there are multiple parties involved, the remote platform could eavesdrop on the communication between the application and accelerators, or the accelerator itself may be a deceiver stealing information. In another scenario, the platform providers or application developers might exploit the intellectual property inside the accelerators.

The above-mentioned attacks with multiple parties involved can be avoided if the platforms are supported with a proper trust model, and suitable security mechanisms. The FPGAs, in fact, have security features already; however, the trust model they are designed for is outdated. Hence, it does not assure security for the computation in the cloud. For this reason, we first look into existing research that aims at solving the weaknesses of the FPGA trust models and propose improved security features. The related work focuses on secure remote FPGA configuration, protecting Intellectual Property (IP) of hardware designers, and the shared used of FPGAs. In addition to surveying these works, we provide discussions on the ways to improve them, and disclose the challenges which have not been addressed by them.

In Section 2, we first provide a brief history of FPGA accelerated computing platforms. In Section 3, we introduce the attacker capabilities on the FPGA based computation and related concerns. In Section 4, we describe three basic trust models; respectively the conventional model that accounts a product developer and its client, the model that accounts IP protection, and the latest model shaped with the adoption of FPGAs by cloud. In Section 5, we survey the existing research that offer solutions to the shortcomings of FPGA security features, and implement improved trust models efficiently. In Section 6, we evaluate the cited related work, and provide discussions on ways to enhance them.

2 INDUSTRY EVOLUTION

In the past, the integration of FPGAs into CPUs was done mostly for research projects and niche markets, but they recently received greater attention and became more available. Some platforms make use of FPGA development boards supported with plug-in PCI(e) ports. Other platforms prefer piecing CPUs and FPGA together on the same die. Today, both are available with a product range varying from low-cost embedded devices to high-performance computation.

Initially, in 2013 Xilinx offered its CPU+FPGA platform for embedded devices: the Zynq SoC [1] offers an ARM processing system and FPGA based programmable logic on the same die. It moves most of the communication ports to the CPU, making communications easier as software can be supported with libraries and Operating Systems (OSs). Moreover, the interaction between the CPU and FPGA is simplified with an AXI-based [15] communication between the hardware modules and the CPUs. Zynq Multi Processor System on Chip (MPSoC) and Radio Frequency System-on-Chip (RFSoC) are also released, which offer greater processing power and an improved FPGA architecture for high performance heterogeneous and domain specific computation.

Next, in 2014 Microsoft publicly announced Project Catapult [51]. FPGAs and CPUs are incorporated in a node [21], and many nodes are connected to each other both on the CPU and FPGA sides. Therefore, CPUs and FPGAs form two layers, and collaboration on a job can be expanded to other nodes from both layers. So far, three versions of the architecture have been reported. While the project was initially introduced to solve the processing problem of the Bing web search engine, later Microsoft expanded it to other web services, and even provided researchers access to it.

In 2015, Intel announced its acquisition of Altera [3], and its Xeon+FPGA platform that integrates FPGAs and Xeon CPUs. Moreover, HARP and HARPv2 (Hardware Acceleration Research Program) were announced for the experimental use of the designed platforms at an early stage. On the hardware side, Intel introduces the concepts of the blue and green bitstream. Intel provides the blue bitstream; the users' accelerators constitute the green bitstream. Partial reconfiguration is used to make the blue bitstream the static configuration of the FPGA, and load the green bitstream to the dynamically configurable regions. The green bitstream interacts with the blue-bitstream for its communication to software and memory operations. This interaction is handled over an interface named as Core Cache Interface (CCI-P). The blue-bitstream translates CCI-P data transfers of green-bitstream to available physical channels of the platform (e.g. PCIe, QPI), hence Intel calls CCI-P an abstract interface. Intel targets accelerator portability with this abstraction, instead of binding the accelerators to a certain device. Moreover, the Xeon+FPGA platform requires accelerators to implement a Device Feature List (DFL) which describes the accelerator and its features, and describes control-and-status registers to interact with them. Making use of the DFLs, Intel's Open Programmable Acceleration Engine (OPAE) [12] provides an API for accelerator enumeration, access and management. Though only Intel knows the content of the blue bitstream, OPAE is developed open source. With Intel's device abstraction, FPGA acceleration is possible with both the Xeon+FPGA architecture and the PCIe enabled Programmable Acceleration Card (PAC) [35]. As a result, it promises portability to various platforms including other vendors' devices. It should be noted that this architecture is still a prototype, which lacks even a proper trust model and basic security features such as access control, and memory protection.

At the end of 2016, Amazon Web Services announced their cooperation with Xilinx to offer FPGA accelerated computation in the cloud [4]. They connect single or multiple powerful FPGAs to the CPU over PCIe channels. Similar to Intel, Amazon provides a hardware module named as *AWS Shell* in the static region of the FPGA, while the users' accelerators are programmed dynamically in the remaining fabric. The user software should use Amazon's provided API to interact with the accelerator through this shell. This usage model has its restrictions; hence precautions against misuse are required. For example, the system allows only a single user on one instance. There is no platform sharing, so the FPGA is explicitly assigned to a single user even though her/his accelerator might use only a limited area. Amazon has created an ecosystem where the accelerators are bound to its hardware, and applications are linked to the accelerators. Amazon also handles a licensing scheme for the accelerators. A novelty within this ecosystem is the introduction of an accelerator marketplace. Developers can upload their accelerators to it, and users can instantiate these accelerators within their applications upon paying a usage fee.

In the last years, Baidu [5], Huawei [7], Tencent [9] and Alibaba [6, 8] also followed with FPGA acceleration in their clouds. Baidu, Tencent and Huawei cooperate with Xilinx, and Alibaba offers both Xilinx and Intel FPGAs on two different types of CPU+FPGA platforms.

In 2017, IBM announced cloudFPGA [14] by fitting a total of 1024 FPGAs to a datacentre rack. It prefers disaggregation of the FPGAs through a network based access to them instead of tight-coupling with CPUs. As a result, IBM offers these FPGA racks as stand-alone resources for hardware acceleration at large scale in datacentres.

In 2017, Microsoft announced another FPGA accelerated computing project named as Project Brainwave [24]. This project specifically targets the acceleration of Deep Neural Networks (DNN) applications. It is available on Microsoft's Azure cloud [13], which offers to users a list of accelerators for their machine learning applications, rather than accepting custom accelerators from users.

In 2019, Xilinx introduced its new device class named Adaptive Compute Acceleration Platform (ACAP), and the first device of this class named Versal. It integrates high-performance ARM cores, with an improved FPGA fabric, and with newly introduced vector processors for big data and artificial intelligence applications. In addition, it claims almost ten times faster configuration than current FPGAs, which is a huge step towards the time-multiplexed use of hardware acceleration. Microsoft already announced that these devices will be deployed in its Azure Cloud.

The developments mentioned in the last two paragraphs might be interpreted as a sign of transformation in the hardware compute platforms, which pays more attention to domain-specific acceleration. The Xilinx's Vitis platform, which offers libraries for accelerating applications of various domains (e.g. artificial intelligence, video coding, data analytics, etc.) with the heterogeneity of CPU and FPGA, supports this hypothesis. Hence, we may witness the evolution of FPGAs with custom ASIC designed for accelerating predefined classes of applications.

In addition to triggering the industry giants mentioned above, the cloud based FPGA services give rise to start-ups, e.g. Accelize. Their business model involves designing custom accelerators for customers, and supporting third-party developers to offer their accelerators to the clients of Accelize. Later, they deploy and manage the applications on one of the above named cloud FPGA infrastructures. Moreover, their solution encapsulates the accelerators into a wrapper, which implements a licensing scheme, and enables accelerator designers to meter the usage of their products.

From this short history, it is clear that there is a big change in computer architectures for heterogeneous and domain specific computation. Fixed CPU architectures are being extended with adaptable architectures that tailor the hardware to the application. Nevertheless, most of them still lack even the basic security features. Hence, now is the right time to discuss trust, security and prepare measures.

3 ATTACKER CAPABILITIES

The literature reports plenty of attacks against FPGA bitstreams to clone their implementation, or to steal designs with reverse engineering. Besides, the bitstream can be used to manipulate the designs for spoofing attacks, or attaching a Trojan to them. Therefore, FPGAs have long been supporting bitstream encryption to avoid plaintext access. However, attacks to obtain an FPGA's encryption key also exist. An overview of basic FPGA security problems and proposed countermeasures can be found in [34, 60].An example attack for key extraction is shown in [?], which uses power side-channel analysis. Moreover, a more recent attack [44] extracts the key using thermal laser stimulation without even decapsulating the FPGA's package. In addition, physical access to the FPGAs enables voltage glitch and laser fault injection attacks, which are described in detail in [20]. Bitstream manipulation is another powerful attack vector. A well-structured summary of bitstream-based attacks is given in [56]. These attacks include e.g. Trojan insertion and fault injection. For the former, [57] shows an example attack that manipulates the bitstream of a security USB flash drive's FPGA to mount a Trojan. For the latter, [56] presents an automation tool that searches for cryptographic hardware implementations in bitstreams, and manipulates them for fault attacks revealing secrets. Above given are only a few examples of the many attack vectors through side-channels, fault injection or bitstream manipulation. We mention them to draw attention to the attackers' capabilities.

The above given attacks have been developed for the old FPGA use-model: the victims are hardware developers who deliver their design to end-users, and the attackers are users who have physical access to the FPGAs. However, the FPGA accelerated cloud computation is changing the use model drastically. The cloud platform providers appear as new entities in the use of FPGA acceleration. As they hold the FPGAs, they have the biggest responsibilities. They should be trustworthy to the developers. Otherwise, the platforms loose credibility and financial benefits. Privacy preserving cloud computing should also be of high concern, because FPGA acceleration often targets critical applications e.g. big-data and machine learning. The privacy concerns increase drastically depending on the processed data, because companies or governments may satisfy their curiosity by peeping at them. Moreover, in the case of FPGA accelerated cloud, all the known attack examples and physical access to the devices could make a platform provider a very powerful and insidious attacker against both accelerator developers and end-users. The providers' access to the programming channels, e.g. JTAG, gives them powerful and easy-to-deploy attack vectors [53].

A malice may also come from the accelerator developers who may turn upon the end-users by performing remote attacks, e.g. with Trojans or side-channels. First of all, neither Intel's Xeon+FPGA platforms nor Amazon's F1 instances offer any memory protection from attackers capable of using FPGA. The FPGA can access any memory location without any restriction, but this could be fixed with the utilisation of IOMMU in the future. However, there are attacks which exploit the memory-architecture using the FPGAs. For example, JackHammer [67] employs a rowhammer attack from FPGA to system memory on the Intel's platforms. The attack yields successful bit-flips in the victim's data. In addition, the authors also explore potential cache side-channel attacks with FPGA's access to CPU's last-level cache through PCIe.

Although the multi-tenant use of FPGAs has not started yet, attacks are made ready. In a recent tutorial [48], a malicious accelerator's ability to perform power hammering attacks is demonstrated. The attacker uses ring-oscillator based hardware constructions to waste excess power, and causes faults in other accelerators on the same FPGA, as the power supply circuitry is shared by them. This attack is powerful that it is undetected by DRC rule checks enforced by the cloud service providers. Similarly, a remote power side-channel attack [71] is shown to be capable of performing a power analysis attack by obtaining the power traces of accelerators. The attack again enables malicious accelerators to launch fault attacks on others. Another remote side-channel attack [43] is demonstrated on the multi-tenant use of FPGAs. In addition, there are attacks [31, 52] capable of stealing information from a victim accelerator through the interference between neighbouring two wires used by the victim and attacker. For mounting such attacks on the cloud, its infrastructure has to be known, i.e. to identify the shared resources. However, the reverse engineering of the infrastructure is tough, because the cloud service providers prevent the use of FPGA's unique identification. However, [59] shows an alternative method based on DRAM based Physically Unclonable Function (PUF). This method has already been used to learn how many unique FPGA instances Amazon has in a specific region, and what is the chance of making a victim use the FPGA which has recently been used by an attacker. More of these attacks against cloud FPGAs and their countermeasures can be found in the recent survey of Jin et al. [38].

Another attack surface is the attackers' impersonation of a reliable developer or its accelerator. For example, an attack can be mounted with a counterfeit accelerator for a cryptographic algorithm, which advertises itself as a reliable design. Such an attack is very easy e.g. with the existing Intel's Xeon+FPGA platform because accelerator developers pick their unique accelerator identifiers themselves, which can be a copy of another accelerator.

It can be foreseen that the attacks over FPGA accelerated cloud will not be limited with the changing role of involved entities, but new attack surfaces will be studied. For example, the attached CPUs employ a channel to configure, debug and communicate with the FPGAs. Hence, the trustworthiness of this channel is an important concern. A malicious or compromised design can be used to sniff or spoof the communication between accelerators and their respective software counterparts.

To explain the security problems better, we evaluate the existing trust models in the following sections, and present the related work that proposes solutions to fix their shortcomings. Next, we will build upon these related works, so that they become applicable for the FPGA accelerated cloud. We will discuss refinements to improve them further.

4 TRUST MODELS

In this section, we present the existing trust relationships between the entities involved in the FPGA use. In fact, today's FPGAs are produced with a fixed trust model, and their security features are designed for it. However, it is no longer valid. For example, the model is not sufficient for a modular hardware development approach allowing the use of IP blocks from third-party developers. Besides, it is also unsuitable with the use of FPGAs in the cloud, which introduces accelerators from third-party developers. In this section, the old and incapable trust model will be introduced at first. In addition, the security features integrated into the FPGAs for this model and their shortcomings will be explained in detail, because they are the building blocks to implement improved solutions. As a second step, the trust model involved in the IP core based design will be described. Thirdly, the model shaped by the FPGA accelerated cloud and its deficiencies will be given. In Section 5, related works that offer improvements to each of these models and their implementations will be introduced.

4.1 Worn Out FPGA Use Model

The longstanding trust model involves only two entities, a product developer and his client, as shown in Figure 1. In this model, the product developer develops an end-user device, which employs an off-the-shelf FPGA. The developer should protect his intellectual property, which is the bitstream programmed to the FGPGA, from piracy. Therefore, he encrypts the bitstream, tieing it to a specific FPGA. The client can be a malicious receiver of the product trying to copy it. This model does not provide any protection if the developers wish to deliver their design without a target FPGA. In addition, the model protects only the product developer, concerned from malice at the client side. However, the client needs to trust the developer, because of his decision to obtain the developer's product. This trust model is enforced by the FPGA manufacturers with the security primitives integrated into the current FPGAs. Though it prevents the designs from being stolen from FPGAs, its benefits are restricted to very specific use cases. In particular, this model is not suitable for FPGAs in general-purpose computation. Furthermore, it is worse for cloud based shared use, since the separation of the FPGA holder, user, and developer is not supported.

The built-in security hardware of FPGAs enforces this trust model by tying the bitstreams to the FPGA with cryptographic keys. The use of cryptography is required because physically binding is not possible. Since the FPGAs are SRAM based, programmed implementations are volatile on them.

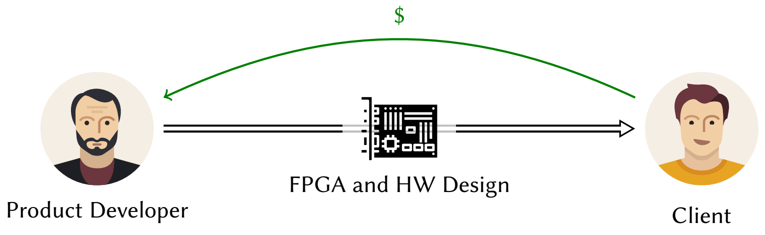

Fig. 1. A hardware product consists of a custom hardware design and an off-the-shelf FPGA device the design is programmed to. The product developers protect their design by delivering it encrypted to the FPGA.

Hence, the bitstreams are downloaded from a non-volatile external memory, e.g. EEPROM, Flash or SD Card. The download is repeated every time the FPGA is powered on or at arbitrary times on demand. That may expose the bitstream to attackers. The cryptography is used to keep bitstreams encrypted in the external memory. In addition, only the bitstreams encrypted with a key specific to the FPGA are programmed. That key is programmed into the devices by blowing eFuses or using Battery Backed RAM. Besides confidentiality, authentication is also supported. In summary, the designers protect their designs by encrypting their design, and shipping the encrypted design together with an FPGA that knows the corresponding key.

Figure 1 shows the product as an FPGA and hardware design programmed to it. The new integrated CPU and FPGA platforms remain using the same trust model. Their platform specific extensions include bitstream programming from CPU side, and security checks for both software on CPU and hardware design in FPGA. For example, Xilinx Zynq integrates the bitstream programming into secure boot. The bitstreams is made a part of the First Stage Bootloader (FSBL), and the device key is used to authenticate and decrypt the FSBL at power up. The FSBL controls the decryption of the bistream and programming it to the FPGA, and hands over the system to next stage bootloader afterwards. As a result, we do not make a distinction between FPGA-only and CPU+FPGA SoC devices when explaining the limitations of the trust model in the following paragraphs.

This model and the underlying security mechanisms are available for long, and their limitations have already created worries. A description of these limitations are given in the following paragraphs. Various designs, which propose fixes to them, are available in literature. Significant examples of them are described in Section 5.

Key provisioning requires physical access to the FPGAs. Therefore, a remote key installation is not possible, since the key is transferred over the programming wires in plaintext. In fact, a promising feature is made available by Xilinx UltraScale and UltraScale+ FPGAs with an internal JTAG port [68], named as `MASTER_JTAG`. Although it allows proramming the keys into devices (e.g. to eFuses or Battery Backed RAM) without physical access to the external JTAG port, it is still not sufficient. An entity who wishes to provision its key to a remote FPGA needs assurances that it is indeed communicating with the target device, or a trusted implementation on the device to program the key to it, and not with an impersonator. Otherwise, a Man-in-The-Middle (MiTM) attack can be mounted. Therefore, when designers wish to deliver an accelerator encrypted, they need to install the decryption key to the target FPGA themselves. In addition, the FPGAs lack features such as key revocation, renewal or establishment. This certainly prevents IP protection when incorporating FPGAs into general-purpose computation platforms, because the FPGA will travel. Besides, its reuse and shared-use by multiple developers are hindered.

There are also concerns with respect to protecting the bitstreams from being overwritten or read back, after they are programmed to the FPGA. The read back of the programmed bitstream may give plaintext access to it. For example, Xilinx FPGAs have multiple channels with different priorities to access the configuration module. The internal channels have the lowest priority, while the JTAG as an external channel has the highest. The JTAG has rich capabilities; therefore, it can be used for mounting various attacks against the programmed bitstream [53]. In fact, both Xilinx and Intel FPGAs offer options to disable the JTAG. One of these options is to disable it through the JTAG interface itself. The second option allows the RTL code to disable it for Xilinx FPGAs. However, these options do not help FPGA accelerated cloud, because they cannot be controlled by partially reconfigured hardware, which is where the accelerators sit. In addition, the accelerators do not have a mechanism to verify if the JTAG is disabled.

One may argue that the use of partial reconfiguration offers a solution to disable the external configuration channels, explained as follows. Partial reconfiguration divides the FPGA fabric into at minimum two parts, one is called the static region and the rest is dynamic region(s). The static region is used to configure the rest. As a result, the static configuration is allowed to reserve the bitstream operations exclusively to itself. Hence, an accelerator can be protected if programmed internally to these regions, by a trusted static configuration. However, that is a deficient solution to achieve complete security. First, it creates a challenge to verify if the static configuration is trusted. Indeed, the FPGAs do not allow the partially reconfigured hardware (i.e. accelerators) to identify or authenticate the static configuration. Secondly, it is always possible to reconfigure the entire FPGA via the external configuration channels. This can replace a trusted static configuration with a counterfeit one immediately after it is programmed. That is a big security concern, because the static configuration has access to the accelerator bitstreams in plaintext, so that it can program them.

4.2 IP Core Based Design

The rapidly advancing technology of FPGAs has made them a good fit even for very complex designs; however, it became a challenge for a single developer to individually design a complete system. Instead, a switch to a modular development approach was promoted, and the developers adopted that fast. It allows creating a system based on hardware modules referred to as IP Cores. These IP cores might implement complex cryptographic algorithms, high bandwidth network or

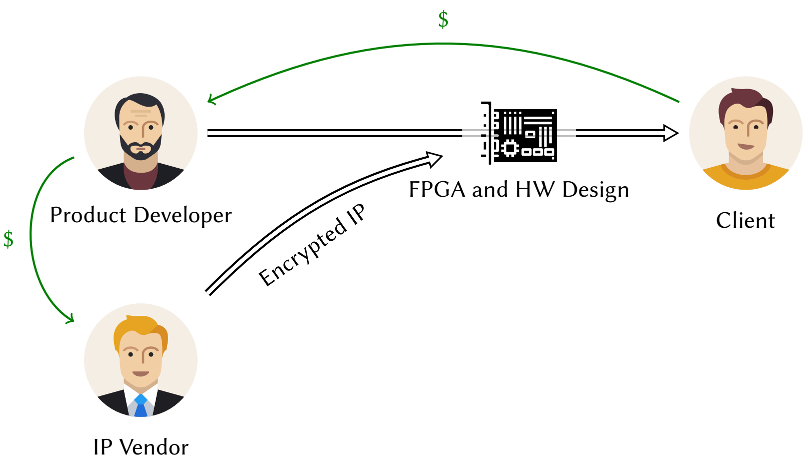

Fig. 2. Product developers use third party IP in their design, and IP vendors wants protecting their IP from the piracy of developers. Hence, they encrypt the IP cores, allowing only the trusted target FPGAs to decrypt.

memory interfaces, video codecs, data processing tools, and more. The IP core vendors look for selling their cores to product designers, who can use the cores in their hardware design. However, selling an IP core is hard if it is a plaintext hardware description, because, a malicios product developer can steal it. Hence, they prefer delivering the IP encrypted, preventing the product developer from seeing its content, but still be able to use it.

In the trust model, the IP vendor appears as a new entity, which is illustrated in Figure 2. The model does not always involve a client. The product developer and client can be the same entity, when developers create an application for their own use. The IP vendors protect their IPs by delivering them encrypted, the decryption of which is only possible by the specific target FPGA.

The IP Core based development has caught on by developers; however, the security and IP rights protection problems have also been brought up. The two major concerns are as follows. First, some of these cores may demand a license fee, but the developers would enjoy using them for free. Hence, they may try stealing or copying them, if the cores are delivered in plaintext. Therefore, IP rights management is needed to protect the cores. Secondly, a developer may receive a counterfeit or malicious IP core. Unfortunately, the FPGA security features are not capable of offering solutions to these problems. Therefore, various works have been proposed to solve these problems, or to improve the earlier solutions. These solutions often involve a trusted authority, who makes the secure transfer of IP cores to FPGAs possible. Significant examples of these work will be introduced in Section 5.

4.3 FPGA Accelerated Cloud

The adoption of FPGAs into cloud computing yields a new trust model at the helm of cloud service providers. This model includes the entities shown in Figure 3. This model the CPU+FPGA platforms are common, while FPGA-only cloud computation is disregarded. This section introduces first the new model with the involved entities and their interactions, and then describes the concerns with it.

The involved entities on an FPGA accelerated cloud platform are as follows. The *platform providers* are the entities who own FPGA accelerated computation platforms and let clients run their software and hardware on it. Amazon and Alibaba are examples of such platform providers who offer remote access to their platforms. Any organisation can compete with them with a custom platform integrating commercial CPU and FPGA devices. The *accelerator developers* create

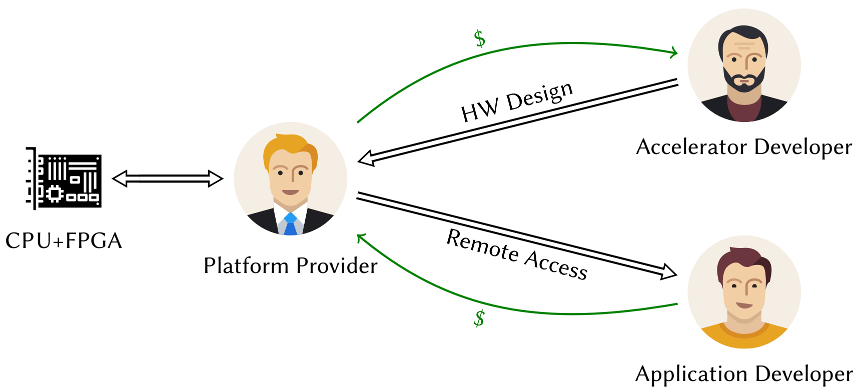

Fig. 3. Platform providers do not trust accelerator and application developers. The developers need to trust to the platform provider.

hardware accelerators that execute specific operations fast on these platforms. The developers look for financial benefits from their accelerators. The platforms' online marketplace could become a medium making their accelerators easily accessible. As another option, they may offer the accelerators to the public for free by sharing them with the open-source community. For example, research projects may prefer this. The *application developers* are the entities who prefer these cloud platforms for hardware-accelerated computation. An accelerator and application developer can be the same entity who prefer cloud over working with a local device. The FPGA accelerated cloud can free developers from the cost of ownership, and make them benefit from the platform provided tools. Similarly, the platform provider can also be an accelerator developer, who offers various accelerators to attrack application developers, e.g. Microsoft Azure.

The trust model of FPGA accelerated cloud introduces significant changes to the previously explained models. The platform providers trust neither accelerator nor application developers. For example, a malicious hardware developer may implement high power consuming hardware that may damage the FPGAs. Yet, the platform providers have financial benefits from marketing the platform. As a result, the accelerator developers might be required to deliver their accelerators to the provider without any encryption, so that the platform providers can verify if the accelerators are implemented according to rules. Hence, the accelerator developers are required to trust the platform providers. A malicious provider can abuse this, and steal the contents of the delivered accelerators, and advertise them on the marketspace as alternatives to the originals. The application developers have to trust both platform and accelerators, if they wish to work with them. A malicious platform can sniff the communication between the applications and accelerators. In addition, a malicious accelerator may try to manipulate the memory space of applications that establish connection to them.

The platform providers protect themselves from the accelerator and application developers by making use of partial reconfiguration. They load the static region with a base-design, e.g. the blue-bitstream for Intel and the AWS Shell for Amazon as mentioned in the Section 2. These base-designs provide partially reconfigurable regions for the accelerators. This is preferred instead of setting accelerators free on the FPGA, which would give them access to the critical platform interfaces such as PCI, QPI (i.e. high performance communication interface connecting IO devices to processor and memory). Restricting the full access is also useful to avoid the imperfect use of these interfaces. Therefore, a base-design offers a simple and restricted interface to the accelerators. It further simplifies the hardware-software interaction with an API for the software parts of the applications. In addition, it is useful to protect the applications from the accelerator developers, e.g. by preventing the accelerators from accessing the critical memory sections.

The platform providers may prefer a platform that comes with a base-design such as Intel's, or they may build a custom platform and implement their own base-design as Amazon does. To the best of our knowledge, some providers (e.g. Alibaba, Huawei, Tencent) typically offer their platforms only to whitelisted customers. We suspect that this decision is an outcome of not having a full-fledged base-design yet to protect the platforms from potential malice. The base-design is a black-box implementation to both the accelerator and application developers. They need to trust the base-design developer, and use it for the interfacing between hardware and software.

The above described trust model and base-design oriented security protects only the platform providers as they administer them. In addition, no feature provides the developers with any protection from potential malice or abuse. Application developers have to trust the cloud service providers, if they decide using their infrastructure. The FPGA accelerated cloud services require them to also trust the accelerator developers, if they do not come up with their own designs. Even if the platform can be trusted, an accelerator might have implemented a malicious hardware.

5 SURVEY OF RELATED WORKS

The previous section introduces three fundamental trust models established for various needs of the FPGA based computation. It describes the limitations of the FPGA security features to protect the hardware IP from untrusted entities, because they are designed with an outdated trust model. This section highlights significant examples from the literature, which aim at addressing better trust models, and offering security implementations for them. The work is divided into the following groups: remote FPGA configuration without revealing the hardware IP, modular hardware development with third-party IP blocks, integrating FPGAs with CPU based computation, and shared FPGA use in the cloud. When combined, these are fundamental to establish trust in FPGA accelerated cloud. After this section, a summary of these work and their features will be given in Table 1. In addition, discussions on how to benefit from these examples on FPGA accelerator cloud, and required improvements will be provided.

The descriptions below assume that the reader is familiar with encryption algorithms, digital signatures and public key infrastructures. Otherwise, the Handbook of Applied Cryptography [50] can be consulted for an introduction.

5.1 Secure Remote FPGA Configuration

This research proposes various designs that improves the FPGA manufacturer provided security features. A majority of these work aim at providing secure updates to remote FPGAs. Some of them explore methods to make an FPGA available to various hardware developers instead of tying them to a single developer. Similarly, some of them propose various licensing strategies rather than a lifetime use permit. Significant examples from the literature are given below together with a short summary.

5.1.1 A Protocol for Secure Remote Updates of FPGA Configurations [28]. proposes to use one FPGA configuration to securely receive the bitstream of an updated configuration from an update server. The configurations assign each device with a unique ID and cryptographic key. It is a symmetric key as a secret shared between the FPGA and the server. The server uses the ID and key for delivering encrypted bitstreams to a specific FPGA over an insecure communication channel. The scheme also provides a remote attestation mechanism, which proves the state of an existing configuration and the update process. The proposed update server can be considered as a Trusted Third Party (TTP) assigning each FPGA with a cryptographic key and configuration bitstream. Hence, the TTP can program various hardware designs to the FPGAs on demand. This solution can be implemented on the current FPGAs, without requiring modifications on their security hardware.

5.1.2 FPGAs for Trusted Cloud Computing [29]. proposes a variant of the previous work, extended with partial reconfiguration. It replaces the initial bitstream with a static configuration. Partial configuration regions are reserved for user design, while the static configuration implements a communication interface to receive them. For the security of this communication, it implements a Public Key Infrastructure (PKI) based on keys provided by a TTP before the FPGAs are deployed in the cloud. The PKI is used to negotiate a symmetric session key for delivering encrypted bitstreams. In addition, the bitstream of the static configuration is encrypted to the target FPGA device for protecting it and the PKI it contains. This solution can also be implemented on current FPGAs.

5.1.3 Fasten: An fpga-based secure system for big data processing [33]. allows users to encrypt their bitstreams to a specific target FPGA on a cloud platform. The platform providers are mistrusted; hence the cryptographic keys of the target FPGAs are obtained directly from their vendor. To make the proposed scheme possible, each FPGA is equipped with a public-private key pair generator based on a PUF. The generated private key never leaves the device. However, the vendors record the public keys of all FPGAs in a database, and associate them with a unique identifier given to each device. The platform providers inform their clients with the identifier of the FPGA assigned to their use. The clients obtain the specific public key of that FPGA from the database maintained by the FPGA vendor. Implementing this mechanism requires PUF based key generator, which is available on MicroSemi FPGA, but not on Xilinx and Intel FPGAs.

5.1.4 SACHa: Self-Attestation of Configurable Hardware [64]. is a trusted computed solution for FPGAs inspired from SMART [30]. The proposed model introduces a prover-verifier relationship. The verifier installs the FPGA device to a remote location, and transfers hardware designs to it over a network at run time. The prover is the FPGA, which receives the updated configuration and proves its integrity to the verifier. As a result, it protects from both a malicious device that lies to the verifier about the state of its configuration, and a network level attacker's modifications on the transferred bitstreams.

SACHa divides the FPGA into static and dynamic configuration regions leveraging on partial reconfiguration. These regions are respectively for the configurations of the proof creating logic and the application. The verifier fixes the trusted proof creating logic to the FPGAs' static configuration before deploying the devices to the field. A PUF based key generation logic is employed in the FPGA for the cryptographic authentication of the proofs. Hence, key storage is avoided. However, the generated key is submitted to the verifier before deployment, so that the verifier can authenticate the proofs created with it. Moreover, the verifier authenticates both configurations, so that a potential counterfeit proof creating logic could be detected. This solution can be implemented on the current FPGAs.

Above given work exemplifies the attempts to secure the remote FPGA configuration. Variations of these work exist in the literature. Some of these [25, 26] consider eliminating network level attacks, e.g. against replay attack to downgrade the system configuration to an older version. Some [63] consider a secure storage and compression of bitstreams at the remote FPGA configuration. In contrast, the FPGA accelerated cloud can benefit from the CPU resources to handle such and similar worries. Therefore, such related works are omitted in this section.

The first two examples given above are early solutions to securely configure remote FPGAs. However, they are still effective, because the FPGA security features still account an old trust model as described in Section 4.1. They depend on a TTP for provisioning cryptographic keys into the FPGAs for secure remote access to them. In contrast to them, Fasten (5.1.3) does not depend on a TTP who shares a secret with the FPGAs to securely update their configuration. However, it demands an active role of the FPGA vendor in the proposed scheme. In comparison, SASCHa does not even require the authority of the vendors, but it demands the owners to provision their FPGAs to the remote field. Lastly, these work propose to have one trusted FPGA configuration to receive and load user applications to the FPGA. This configuration could be considered as an early version of the base-design the FPGA accelerated cloud uses today.

5.2 IP Core Based Development

This section provides various examples from related work, which focus on security of IP cores that system developers use as building blocks for their designs. These work aim at protecting the IP cores from piracy, and licensing their use. Unfortunately, they are not adopted in real life applications. Developers and companies adopted neither of them, because their protection does not extend to the development phases. They do not allow the system developers to involve the IP cores in the simulation and debug steps. They protect the IP cores at the bitstream level; hence only allow their use by the end designs. However, we give these work a second chance on the FPGA accelerated cloud, for which the bitstream level protection is acceptable and already in practice. In this setup, accelerators are the IP cores re-delivered as bitstream, and their functionality includes direct communication with the software counterpart than with each other. As a result, extending the protection to the development phases is not required. Therefore, we study the proposed IP core protection mechanisms with the purpose of re-using them for accelerators on the FPGA accelerated cloud.

5.2.1 Dynamic Intellectual Property Protection for Reconfigurable Devices [32]. is an early work that proposes a key establishment scheme between the FPGAs and IP vendors. The proposed scheme asks FPGA vendors to burn a unique key into each FPGA, and prepare a hardware configuration encrypted with that key. This initial configuration provides the FPGAs with a unique ID, a public and secret key pair, and implements a key establishment scheme. FPGA holders use this configuration to make their FPGAs establish a key with the IP vendors when requesting a license to their IP. During this phase, the FPGA holder forwards the messages of FPGA and IP vendor to each other. Upon success, the FPGA loads the established key to a secure key storage attached to the FPGAs. Afterwards, the FPGA can decrypt and load bitstreams received from the corresponding IP vendor. This solution requires modifications on the current FPGAs for secure storage of the established keys.

5.2.2 Protecting multiple cores in a single FPGA design [27]. extends the above work for protecting multiple IP cores. These extensions involve repeating the key establishment scheme with various IP vendors, for deriving a key to use their IP cores. This design requires a bigger secure key storage to support the plurality of IP vendors. In addition, it benefits the partial reconfiguration for dividing the FPGA into unequal regions intended for different IP cores. Besides, it stitches together the bitstreams of these regions, accommodating them with additional information so that the FPGA knows which key should be used for each region.

5.2.3 SeReCon: a Secure Dynamic Partial Reconfiguration Controller [40]. protects IP with a Root-of-Trust (RoT) hardware for FPGAs. In this setup, the RoT hardware is an FPGA configuration that targets the static region. The RoT is associated with an ID for generating device specific cryptography keys for IP protection, and confidential data storage. The keys are derived internally, and never leave the FPGA. Moreover, a trusted authority is introduced to confirm the key generation, and certify the public keys. IP vendors obtain the certified public key of their target FPGA from the authority, and encrypt their IP with it. The RoT hardware decrypts the received IPs with the corresponding private keys, and program them. In addition, the FPGAs are extended with a bitstream authentication hardware, which uses the device keys to authenticate the RoT configuration at each time it requests access to the keys. As a result, even run-time manipulations on the RoT can be detected.

5.2.4 A pay-per-use licensing scheme for hardware IP cores in recent SRAM-based FPGAs [47]. proposes a metering hardware for charging the IP core use. It introduces a scheme that relies on a TTP who receives the FPGAs from their vendors, equips them with unique cryptographic keys, and provides a metering hardware tied to them with the same key. It also introduces a key management scheme, in which IP vendor keys are registered by the TTP. These vendors deliver their bitstreams encrypted to the FPGAs. The FPGAs consult to the TTP for the decryption keys, for which the messages between the FPGA and TTP are delivered by the FPGA holder. The TTP provides the IP decryption keys to FPGAs encrypted under the FPGA specific key, so that the FPGA holder cannot learn it. Implementing this design does not require changes on current FPGAs.

5.2.5 A Pragmatic Per-Device Licensing Scheme for Hardware IP Cores on SRAM-Based FPGAs [70]. places the FPGA vendors between the IP vendors and product developers for practicability. In addition to simplifying their communication, it allows the FPGA vendors to manage a catalogue of IPs for the developers. Although the proposed scheme relies on the FPGA vendors for managing such a marketplace, it does not place unquestionable trust into them, and prevents their access to the IPs. The FPGA vendors receive the developers' requests to use an IP, but forward these requests to the IP vendors.

The proposed scheme also allows the confidential transfer of the IPs from their vendors to the FPGAs. The vendors prepare a core installation module to decrypt and program their IPs according to the requirements. The IP vendors make the trusted FPGA vendors encrypt this module to the target FPGAs. For making this encryption possible, the FPGA vendors are asked to provision unique secret keys into the FPGAs. The scheme can be implemented on the current FPGAs without requiring changes.

5.2.6 Secure Local Configuration of Intellectual Property Without a Trusted Third Party [41]. is a recent work which is concerned with placing trust into any entity. It employs the integration of FPGAs with CPUs to associate the bitstreams with dedicated secure programming software. Specifically, it proposes to deliver an encrypted IP together with a software application, which will run on the CPU of the platform. The CPU decrypts the IPs and programs the FPGAs. The use of a hardware-assisted software (HAS) protection USB dongle is proposed for protecting the decryption key by tying it to the application. This scheme can be implemented on the current CPU+FPGAs SoC devices, such as Xilinx Zynq.

The above given works are significant examples that aim at extending the FPGA security features to promote IP core based modular hardware designs by preventing IP piracy. In summary, they rely on device specific keys hidden from untrusted device holders to prevent them from access to the IP bitstream in plaintext. For allowing the IP vendors to authenticate the FPGAs, they require a trusted authority to associate the devices with secret keys. Though some of these are early works, their proposed use models have similarities with the FPGA accelerated cloud platforms. For example, [40, 47, 70] proposed to use the static configuration to perform security checks on the IPs that target partially reconfigurable regions. Additionally, [70] already looked into an IP marketplace managed by an untrusted party. In addition, it incorporates a different use of base-design compared to all other works. The base-design is picked by the IP vendor to protect its IP with, and delivered to FPGAs together with the IP. Lastly, [41] proposes to benefit from the hardware-software co-design nature of the platforms to achieve protection. A detailed evaluation of these solutions and required improvements to aid FPGA accelerated cloud will be provided in the discussion section.

5.2.7 IEEE P1735 - IEEE Recommended Practice for Encryption and Management of Electronic Design IP [2]. is a method that Electronic Design Automation (EDA) tools use to support development with encrypted IP cores. This solution introduces and extra trusted party, namely the EDA vendor or tool. This method allows to encrypt the RTL or netlist of an IP, which becomes decryptable only by the EDA tools running on the system developer's computer. Although the developers cannot get access to the encrypted IP, they can involve the IP in the development of their design. For example, the input and output behaviour of the IP are observable in the simulations. Besides, they are given control over the end product, e.g. different FPGA targets, various synthesis/implementation strategies, piecing it with additional hardware modules. To benefit from this approach, the IP Core vendors should trust an EDA tool, encrypt their core and make the decryption key accessible to the EDA tool. The IEEE P1735 has already become a standard supported by the major FPGA and EDA tool providers. However, Chhotaray et al. [23] pointed out its vulnerabilities which even allow a complete recovery of the protected design, or attaching a hardware Trojan to it.

5.3 FPGA Accelerated Computing

For getting the most out of reconfigurable devices, some researchers are interested in developing a hardware supported OS. These work often divide the computation between software executed on a CPU, and hardware running on an FPGA. These two are supported with an operating system for their synchronisation, scheduling and file access. In the paragraphs below, some examples of these work are summarised.

Please note that these are early works which only integrate software computation with hardware, ignoring the trust and security considerations. They are presented below to illustrate the different ideas to benefit FPGAs in computing, but will not be taken into account when evaluating the security aspects of proposed work in the later sections. However, the base ideas these work introduce are taken over by the recent studies that target FPGA accelerated cloud computing, and these work will be elaborated in the next section.

5.3.1 A Runtime Environment for Reconfigurable Hardware Operating Systems [65]. introduces a runtime environment for computation in hardware. Its major contributions focus on managing the FPGA resources for the partial reconfiguration of hardware computation blocks. Instead of offering a fixed size for all blocks, it divides the reconfigurable region into adjacent slots. It also allows varying size computation blocks by reserving them the number of slots they demand. In addition, they implement a bus based communication interface for the slots to communicate with each other or with the host CPU. However, the security of this communication or resource sharing is not elaborated.

5.3.2 A unified hardware/software runtime environment for FPGA-based reconfigurable computers using BORPH [55]. presents an OS that accounts for computation in both hardware and software. It introduces the *hardware process* concept for parts of applications handled in the FPGA. Besides, it extends the Inter Process Communication (IPC) interface of POSIX for the communication of hardware processes with each other and with software processes. This interface allows file sharing, pipes, signals, and message-passing. Hence, the hardware processes also have access to system resources e.g. the file system, standard data input and output.

5.3.3 ReconOS: Multithreaded programming for reconfigurable computers [45]. proposes an OS that introduce hardware threads which cooperate with conventional software threads. The OS simplifies the inter-thread communication by associating each hardware with a *delegate* software thread. These delegates represent their corresponding hardware, enable their synchronisation to each other and to the software threads with the support of semaphores, mutexes, message boxes, and shared memory use.

5.3.4 FUSE: Front-end user framework for O/S abstraction of hardware accelerators [36]. can be considered as similar to ReconOS, developed with slightly differing design decisions. It eliminates the need delegate threads, besides introducing an interface between hardware threads and OS that is customisable for specific application. Unfortunately, it assumes a simpler memory model by ignoring the memory virtualisation, which is a foundation for an OS to isolate applications and offer security.

Above given works focus on achieving a greater computation power by making use of hardware programmability, but lack trust and security considerations. We may assume that they rely on the basic security mechanisms (e.g. access permissions, memory isolation) of their proposed runtime or OS. The next section provides improved versions of these works which specifically target cloud computing.

5.4 FPGA Accelerated Cloud

CPU or GPU based cloud services allow more than one user on the same platform with the isolation of resources reserved to them, by means of virtualisation. However, the existing FPGA accelerated cloud services do not address similar isolation primitives. Instead, they allocate a whole FPGA to a single user, even though the users would utilise only a very small portion of the available FPGA resources. In the subsections below, we list examples from related work that aim at the secure and shared use of FPGAs in the cloud.

5.4.1 Enabling FPGAs in the cloud [22]. demonstrates a complete software stack which extends the virtualised execution based user isolation in the cloud with FPGA resource access. It divides the FPGA resources into slots, and gives users access to these slots from their virtualised execution environment. Hence, it invests into extending the memory isolation primitives of virtualisation for FPGA resource accesses.

The design can be implemented on existing FPGAs, but it suffers from creating a unified pool of FPGA resources for creating flexible slots. To overcome this problem, the proposed design asks from clients to submit the source files of their designs to the cloud service providers. The provider compiles each design for all possible slot targets, to make them slot independent. Bothered with the complications of this method, the authors demand to have a unified FPGA bitstream structure that will allow accelerator portability to different FPGAs.

5.4.2 Virtualized Execution Runtime for FPGA Accelerators in the Cloud [16]. introduces a runtime manager in the FPGA to mediate between the software threads and their hardware accelerators. This manager is software that runs on a soft-core processor implemented in the FPGA. The applications interact with the manager for requesting FPGA acceleration. The manager receives the requests, and serves them with workload aware FPGA resource management. For example, it programs the accelerator to a pre-defined partially reconfigurable region when the region becomes available, and later start the execution of the requested operations. In addition, it is responsible of memory management, such as address translation between applications and accelerators. This design can be implemented on the existing CPU+FPGA platforms.

The runtime manager is provided with application specific software, compiled with the developed toolchain for the corresponding applications and their accelerators. This is the worrisome part of the proposed implementation, because the runtime manager executes a code received from users, besides being responsible for preventing malicious user activities.

5.4.3 The Feniks FPGA Operating System for Cloud Computing [69]. is an FPGA operating system designed for making multiple accelerators share the FPGA resources in the cloud. Its three main features are the support of multi-tenancy, the direct access of accelerators to system resources, and a framework allocating the system resources to these accelerators. These features are made possible with a base design that is called an OS in this context. This OS defines various partial reconfiguration region templates to the accelerators, and loads them to a region with a matching template. It is developed over Microsoft's Catapult platform described in Section 2. In contrast to the hardware-based OS primitives given in Section 5.3, it does not address synchronisation of software applications with their respective hardware accelerators. An implementation of this design is demonstrated on Microsoft's Catapult [21], which has a sophisticated CPU and FPGA integration mechanism as described in Section 2.

5.4.4 Cryptographically Secure Multi-Tenant Provisioning of FGPAs [17]. aims at solving the scalability problem of key management for the multi-tenant use of FPGAs on the cloud. Leveraging on Key-Aggregation and Bring-Your-Own-Key mechanisms, it allows accelerator developers to hand their encrypted bitstreams over the FPGAs, together with the corresponding decryption keys. The keys from multiple users are aggregated into a fixed size secret key in the FPGA. This secret key is used to obtain the corresponding decryption key when the users wish to programme their accelerators. With this solution, the increasing number of users sharing the FPGAs does not affect the required secure key storage size. This mechanism also relies on a trusted FPGA vendor, who will assign each FPGA with unique public-private keys, so that the developers can deliver their decryption keys under encryption to the remote FPGAs.

5.4.5 FPGAs and the Cloud – An Endless Tale of Virtualization, Elasticity and Efficiency [42]. is an extensive work exploring the FPGA virtualisation in the cloud. Part of this work focuses on providing a partial reconfiguration based resource sharing, and a corresponding runtime for allocating these resources to users and applications. This solution is tailored for a use model anticipated for the future of FPGA accelerated cloud. Furthermore, it also considers the secure transfer of accelerators to the FPGAs. The proposed use model and secure bitstream transfers are explained respectively in the next paragraphs.

The accounted use model is slightly different from the model of *FPGA Accelerated Cloud* explained before in Section 4.3. It divides the role of platform provider into multiple entities, which are explained as follows. Datacentre operators construct their infrastructure. Cloud providers rent the devices from the infrastructure and create virtual computation instances on them. Service providers rent these instances, and implement their web services on them. End users enjoy these web services. In this hierarchy, the datacentre operators provide FPGAs. Cloud providers create virtual FPGAs by dividing the FPGA resources into parts. They can program these virtual FPGAs for the needs of either their own or the service providers renting their virtual instances.

Proposing an IP protection mechanism is essential for this model, because otherwise an accelerator developer should be asked to trust the service provider, cloud provider and datacentre operators, all at the same time. For this protection, a subset of Transport Layer Security (TLS) is implemented in the static region of the FPGA, which establishes a secure communication channel to the developers. The availability of FPGA specific public key pairs are assumed for authenticating the target FPGAs when establishing this channel. However, the authors do not describe how the FPGAs receive their private key. Obviously, there should be a trusted party either to assign such keys to the FPGAs, or to keep track of internally generated keys, although the existence of an authority is against the security requirements the authors list.

The above cited works are not the only research for FPGA accelerated cloud, but major examples. The survey of Vaishnav et al. [62] evaluates a few of them with criterions such as performance, scalability, flexibility, and isolation.

Unfortunately, these work only extend known software based isolation primitives to FPGAs. They omit revisiting the trust management for the changing architectures, and addressing the hardware IP protection concerns. In addition, the proposed designs could not be claimed as a complete solution to secure multi-tenant use. As a result, the platform providers allow only one accelerator and application at a time on the massive FPGA devices, no matter how much of it is utilised. Sticking with such an inefficient isolation gives rise to a waste of precious reconfigurable hardware resources. Furthermore, unshareable resources keep the cost of using FPGAs high.

6 DISCUSSION

The secure and shared use of FPGAs is essential to get the most out of them in the cloud. Moreover, there is no solution offering a complete and practical design to the best of our knowledge. Therefore, in this paper we strive for studying the related research focusing on the various use cases of FPGA based computation, and for exploring ways to refine them for the FPGA accelerated cloud. The sections above shed light on the major phases of FPGA based computation, with descriptions on accounted trust relationships and security features. To clearly depict the approaches of various IP protection and security features, examples from related works have been cited and described. In this section, we provide discussions on ways to improve these related work, in addition to revisiting them with their security features and underlying assumptions.

Table 1 summarises these works with their proposed trust assumptions and security features. These works agree on the need of assigning cryptographic secrets into FPGAs for the authentication of target FPGAs, protecting IP of hardware designs, and attestation of the available FPGA configuration. For example, the table shows that IP protection can only be achieved by cryptographically tieing the IPs to specific FPGAs, which requires encrypting them to the FPGAs. Only three of them [33, 40, 41] could offer a solution free from hiding shared keys into the FPGAs; either by generating keys in the FPGA [33, 40] or using an external key device [41]. Furthermore, most works (except [41, 64]) prefer the presence of an authority for the key management. In addition, they perform security checks on accelerators by the use of a base-design that dynamically programs them, or a design that is replaced with them. That is preferred as the FPGA vendor provided security features are incapable for advanced protection mechanisms. In addition, FPGA accelerated cloud with a base-design simplifies the accelerators' communication to the software.

The sections below revisit some of the considerations that related works already studied, and provide discussions on their shortcomings and suggests improvements. In addition, some sections will provide discussion which are omitted by these related work, but essential for the establishment of trust in FPGA acceleration on cloud.

6.1 Base-Design Identification and Authentication

Many of the aforementioned protection mechanisms rely on partial configuration to load the infrastructure and user hardware into respectively the static and dynamic regions. The FPGA cloud providers follow today a similar strategy to this, with the base-design and accelerators. Therefore, adaptations of the related work to the cloud settings is possible. However, the lack of a mechanism for users and accelerators to identify and authenticate the base-design is a concern, which has not been addressed yet. The evaluation of a base-design by any of the accelerators is not possible by the related works, while the inverse is achievable.

The base-design, as the static configuration of the FPGA, is capable of reserving programming channels to itself, as already described in the Section 4.1. These features extend to disabling JTAG and debugging features, which are elementary actions against an untrusted device holder. As the accelerators are not capable of doing such operations, their verification of the base-design appears as a necessary feature. In addition, the plurality of base-designs, their versions, and demands for accelerator portability alleviates the importance of a feature to evaluate the base-design. It gives users the option to demand a trusted base-design from the platform providers. That could be considered akin to various OS options that users are given when they rent a server today. Such options could aid the trust problem, instead of forcing users to rely on the custom implementation of cloud providers. In addition, it may even trigger the use of an open-source base-design such as [37].

Table 1. An overview of the referenced solutions are given with a comparison of their features.

		Protects IP	Ties HW to FPGA	Shares Secret Keys	Uses Base Design	Modifies FPGA	Allows Many Users	Allows Many IPs	Device Holder	Trusted Authority
Remote FPGA	[28] 2009 Drimer et al.	●	●	●	◑	○	○	○	Anyone	TTP
	[29] 2012 Eguro et al.	●	●	●	●	○	○	○	Cloud SP	TTP
	[33] 2018 Hong et al.	●	●	○	○	◍	○	○	Cloud SP	FPGA Vendor
	[64] 2019 Vliegen et al.	●	●	●	●	○	○	○	Anyone	None
IP Protection	[32] 2007 Güneysu et al.	●	●	●	◑	●	○	○	Anyone	FPGA Vendor
	[27] 2008 Drimer et al.	●	●	●	◑	●	○	●	Anyone	FPGA Vendor
	[40] 2008 Kepa et al.	●	●	○	●	●	○	●	Anyone	TTP
	[47] 2012 Maes et al.	●	●	●	●	○	○	●	Sys. Dev.	TTP
	[70] 2014 Zhang et al.	●	●	●	●	○	○	●	Sys. Dev.	FPGA Vendor
	[41] 2019 Khan et al.	●	●	○	◑	○	○	●	End User	None
FPGA Cloud	[22] 2014 Chen et al.	○	○	○	●	○	●	●	Cloud SP	Cloud SP
	[16] 2017 Asiatici et al.	○	○	○	●	○	●	●	Cloud SP	Cloud SP
	[69] 2017 Zhang et al.	○	○	○	●	○	●	●	Cloud SP	Cloud SP
	[17] 2017 Bag et al.	●	●	●	●	○	●	●	Cloud SP	FPGA Vendor
	[42] 2018 Knodel et al.	●	●	?	●	●	●	●	Cloud SP	FPGA v. / TTP

◑ Not partial reconfiguration based. The whole configuration is used to securely receive applications.
◍ Not all FPGAs have to be modified, some already supports required features.

The accelerators cannot evaluate the base-design today, because of a limitation that FPGAs have. They reserve the internal programming channels to the static configuration, so that it can manage the partially reconfigurable regions. However, partially reconfigured regions are not given a direct access to them. That restriction in fact can be considered as a protection mechanism, preventing an accelerator from manipulating the base-design or other accelerators. However, this limitation also restricts the accelerators from verifying the base-design.

Within the cited related works, only SACHa [64] offers a way to verify the base-design, albeit it needs the attestation of the entire FPGA configuration. In addition, Zhang et al. [70] offer an interesting solution, in which the IP vendors deliver a base-design to FPGA holders, for the programming and protection of their own accelerators. However, giving the complete control of the base-design to the accelerators cannot be welcomed. It would take away from platform providers the way to protect their platforms, and the interfaces they developed for simply integrate accelerators with software. As a result, we opt for letting the platforms manage the base-designs the users pick, and allow the users and IP vendors to identify and authenticate the base-design.

6.2 Comparison to Software Marketplaces

The FPGA accelerated cloud platforms, as offered by Amazon's AWS, could be organised similar to the software marketplaces, e.g. Apple's App Store. In the case of App Store, app developers and users have to trust Apple. Developers sign their apps before submitting them to the store. Certificates issued by Apple proves the designers' ownership of the signing key. The users can question their trust to the apps and their developer with the credentials that Apple advertises on the store. If users trust and decide to install an app, their device verifies the app with Apple's certification. Similar to Apple's model, Amazon's FPGA acceleration marketplace relies on the trust on Amazon.

In contrast to the mobile phone app market which has only two major players (i.e. Apple and Google), there are already a significant number of FPGA-accelerated cloud service providers. More may appear in future, if datacentres find it profitable and equip their platforms with FPGAs. Hence, it is hard for users to evaluate and trust each provider. Furthermore, the platform providers may prefer a third party base-design instead of implementing their own. That would resemble Google's Android, which runs on mobile phones from various vendors, but claims Google as the central trusted entity.

Both examples given above solve the trust problem by enforcing an absolutely powerful central trusted authority. A direct adaptation of these models to the FPGA accelerated cloud is a very remote possibility. It requires an authority to earn the trust of the accelerator developers, FPGA vendors, and cloud platform providers. We argue that the trust can be earned, if one promises a scheme to protect the IPs, while restricting the provider from free access to the IP. That prevents a direct adaptation of the software marketplaces to FPGA accelerated cloud, but requires a specifically tailored solution.

6.3 Central Trusted Authority

Most of the referred works rely on a central trusted entity. This entity is either an FPGA vendor trusted by the system developers and IP core vendors, or it is a TTP who is even trusted by the FPGA vendors. We argue that while asking everyone to trust an entity is already hard, a motivation is needed for that entity to earn the trust of everyone. Unfortunately, such discussions are not elaborated in the cited related works.

For long, the FPGA vendors provide security solutions within their devices; however, they avoid hiding pre-shared secrets into them. In fact, proposing a design that suggest the vendors to change their mind and to provision secret keys into devices cannot be considered reasonably practicable. Their proficiency in developing a device for high performance computing should not imply that they can also be good in secret keeping. Moreover, the secret installation is very complicated, since device manufacturing often require working with subcontractors that may learn the secrets.

The lack of device specific secrets limits the practicability of the proposed remote key installation mechanisms of related works. A better solution amongst the cited work is proposed by Serecon [40] and Fasten [33]. They rely on a PUF to generate device specific key pair, the secret of which never leaves the FPGA. Though an active contribution of an authority is required for the certification of the corresponding public keys, at least the secret key installation and sharing is avoided.

Some related works [28, 29, 40, 47] rely on a TTP; however, they lack at providing means for the TTP to participate. The TTPs may be disregarded when they ask for a fee to sign a design, or to keep logs. Instead, involving them in the model e.g. with a role of managing an online IP Core marketplace could introduce sufficient incentives. An obvious incentive would be a percentage share from the license fee of each IP core. In that case, they can be expected to protect the designs to the full extent, as profits will rely on it. For this goal, they may even offer to the cloud owners their custom base-designs.

Accelize might be considered as such a start up trying to address a similar model on its own. They claim a proprietary Digital Rights Management (DRM) solution for the accelerators, which is compatible with various FPGA accelerated cloud platforms. Their DRM wraps the accelerators, protects them, and meters their use. In addition to protecting the accelerators, the developers can also advertise them to the other customers of the company. However, it again introduces a new trusted party, which is the Accelize itself.

6.4 Cryptographic Key Management

As seen in the Table 1, most solutions protect IP using shared secret keys. In fact, related works [33, 40, 41] showed that shared keys can be avoided if key generation is handled inside the FPGA. An outlier of them is the key aggregation mechanism of Bag et al. [17], which aims at provisioning secret keys on cloud FPGAs without sharing them with another entity. However, their solution is not perfect, as the keys are still used for tying the accelerators to specific target FPGAs. It is a disadvantage for cloud applications, restricting the cloud service providers from porting an accelerator from one FPGA to another. As a result, the providers lose the ability to manage their own resources. As a simple solution, the proposed key establishment schemes could be re-executed each time the FPGA target of an accelerator is changed, or a new target is added. That could simply work when the clients work with their custom accelerators. However, if the clients prefer to use an off-the-shelf accelerator, the key establishment would rely on the availability of accelerator vendors. On the one hand, no cloud service provider would prefer it, as it could result into limps in their service. On the other hand, an offline solution is a disadvantage for the accelerator vendors. It would leave the vendors out of the scheme, e.g. hiding from them how many times their accelerators are used. As a solution, the TTPs could be involved as highly available entities who are responsible of the corresponding key management.

A solution could be achieved with the use of a proxy re-encryption algorithm [66], which works as follows. Accelerator vendors send to cloud service providers their designs, encrypted under their own key. Next, the accelerators are made available to the providers, but they cannot decrypt. However, they are the proxies able to alter the encryption so that the target FPGA can decrypt. This alteration is only possible with a re-encryption key derived from the keys of the accelerator vendor and the target FPGA. The cloud provider could obtain the re-encryption key from a TTP, who should also issue keys to the FPGAs. As a result, the cloud providers cannot decrypt the accelerators; however, they can still load them to a target FPGA they pick. This solution protects the accelerator confidentiality, gives the vendors control over their accelerators via the TTPs, solves the availability problem, and allows the providers to pick target FPGAs freely. We provide an elaborated sketch of this solution in our recent report [61].

A concern regarding privacy issues may show up when accelerators are encrypted to a specific target FPGA. For example, the unique identification of cloud service providers' devices is a concern that has already been expressed for trusted computing when devices used their unique keys for attestation. However, solutions are found, and they can be adapted to the FPGA accelerated cloud computation. These solutions leverage on the group signature scheme introduced by the Direct Anonymous Attestation (DAA) [19] of Trusted Platform Module (TPM), and extended into Intel's Enhanced Privacy Identifier (EPID) [18]. It associates the devices with a group, e.g. the devices of a particular cloud service provider. A TTP assigns each group with a public key, and each device with a unique private key, employing a group signature scheme. This scheme allows a device to prove its membership of a group, while preserving the individual device identity. Hence, a key establishment can be employed relying on this scheme for allowing the users to establish a key to an arbitrary FPGA of the group. Such a scheme could extend the device registration methods of the cited works [27–29, 32, 47, 70], in which a TTP assigns a public and private key pair to each FPGA.

6.5 Protection at Bitstream Level

Most IP protection works protect the designs at the bitstream level. Therefore, they do not support the system development phases for simulating and debugging interoperation of the IP cores with other hardware blocks. This limitation was their biggest disadvantage, and hindered their use in real life applications, or adoption by the industry. Yet, they have a second chance with the advent of FPGA accelerated cloud.

The accelerator designs can be kept confidential with encrypted bitstreams, but some third party design investigation tools wish to access them in plaintext for analysing their designs. For example, Matas et al. proposed *FPGADefender* [48] for detecting power wasting malicious hardware constructions. Besides, the accelerators are not always delivered to the cloud as a bitstream, as mentioned in Section 4.3. Intel's Xeon+FPGA platform prefers bitstreams, but Amazon AWS receives synthesised netlist. That enables Amazon to tune the accelerators for the target FPGAs, e.g. to replace the target FPGA, reshape the reconfigurable regions, apply design-rule-checks and placement constraints for it. IP protection is possible at the netlist level with the support of development tools for encrypted bitstream. However, the IEEE recommended standard of this protection is proven insecure as described in Section 5.2. Therefore, refinements are required.

The netlist level protection can also be achieved through the open-source projects that aim at developing hardware development tools for FPGAs, instead of waiting for FPGA vendors to take actions. For example, the ambitious goal of SymbiFlow [58] is to set the open-source standards for front- and backend to hardware development languages with vendor-neutral tooling that would cover all the necessary components for an end-to-end flow.

6.6 Shared Use of FPGAs

The shared use of FPGAs is another lacking feature of today's FPGA accelerated cloud. The service providers assign an FPGA as a whole resource unit to a single user. In addition, these FPGAs are picked from the high-ends of their class, offering a large reconfigurable fabric. The amplitude of available resources makes developers happy by eliminating their utilisation worries and helping them focus only on the performance. Yet, the costs increase, so new constraints appear. In addition, it could be simply considered as a waste of resources when accelerators occupy only a small portion of the reconfigurable fabric. Instead, multiple applications could benefit from a single FPGA, and share the costs. Therefore, the shared use is beneficial for both users and platform providers.

One obstacle to the shared use is the portability of accelerators to different configurable regions in the FPGA. The partial reconfiguration of today unfortunately restrains a design to a pre-defined region. This problem appears with the complexity of finding equivalent regions. The problem gets worse with the growing region sizes. Though the equivalent regions by means of their available resources could be found, they may still not be used interchangeably, because of static configuration wires routed differently through them. Extra efforts are needed to avoid such routing, which demands tricking the EDA tools. Unfortunately, finding equivalent regions can only be simplified by the FPGA technology. The FPGAs should allow dividing the programmable fabric into regions, which are simply an exact copy of each other. Even smaller variations are not tolerable. For example, different directions of the propagation of clock signal may result into observable effects on the timing. In addition, complete isolation of regions should be achievable, so that interference between them and related attacks [31, 52] can be prevented.

Enabling the multi-tenant use of FPGA should not be designed only with a cost perspective, which will make users share the device resources maximally. In addition to designing the best solution for high Quality-of-Service (QoS) for all tenants, potential malice from one tenant to other should be prevented. Otherwise, the shared resources will enable the attacks mentioned in Section 3. As an alternative approach, multiple small but isolated FPGAs that can be clustered on demand could be a better solution than a massive FPGA shared by multiple users.

The related works that focus on FPGA accelerated cloud already account for sharing the FPGAs amongst multiple users, as it is seen in the Table 1. Some assume that equivalent regions could be found [65], and some assume the availability of an accelerator's implementations for various regions [16, 22]. In addition, they propose memory management solutions for addressing the security in shared use; however, they omit protecting the IP of accelerators. Another subset of the related works aims only at IP core protection. Although they are not designed specifically for FPGA accelerated cloud, their cryptographic primitives protect IP vendors already. The cloud use would require improving these work to support both IP core protection, and access control management. For the latter, it should manage whether an accelerator programmed by a certain user is accessible to other users. The file permission checks of modern operating systems can be tailored for the accelerators.

6.7 Attacker Models

A strong attacker model requires a strong security solution. Such a model should not consider only which entities are untrusted, but also a probability that they may unite their powers and mount an attack together. For example, in the work of Maes et al. [47] the roles of IP vendor and system developer can be united to perform a chosen plaintext attack over the TTP. It works as follows. First, the attacker acts as an IP vendor to choose the plaintext by asking a TTP to register its IP decryption keys. These keys are the chosen plaintext, and there can be many of them for multiple keys for various IPs and various versions of them. Afterwards, the attacker acts as a system developer who obtains licenses to these IPs. The TTP sends the IP decryption keys encrypted to the FPGA. As a result, the attacker knows the plaintext submitted to the TTP, and corresponding ciphertext encrypted under the FPGA key. If it can extract the FPGA key, it can obtain IPs of interested vendors, then decrypt and steal them. In order to avoid such vulnerabilities, besides the efforts on picking strong cryptographic algorithms, cryptographic protocols should be carefully designed.

6.8 Implications of Security over Performance

In software, an attacker has many different attack surfaces. For example, s/he can use buffer overflows to manipulate stack or execute return-to-libc attacks. The studies show that cache side channels and speculative execution are also very powerful attacker tools. The operating systems often implement mechanisms to isolate applications better and to randomise this isolation, so that such attacks can be prevented.

The nature of FPGA based computation is highly different from software. The most important difference is that, the intermediate values of a hardware based computation is stored locally in its registers, but not exposed to outside world (e.g. external memory or caches). That prevents sharing these values with CPU or among co-located accelerators. In addition, a hardware design often requires a well defined input/output behaviour and memory access logic. These already eliminate many attack surfaces. Obviously, multi-tenant use of hardware could yield into shared use of resources and could create worries about physical side-channels, such as power use, magnetic interference. However, these can be solved with more straightforwards precautions compared to software, if programmable regions on the FPGA can be isolated into well defined regions, as mentioned in Section 6.6. As a result, an active protection for hiding the secrets of FPGA based computation can be achieved with simpler constructions compared software. Thus, in comparison to an unprotected computation performance, we expect too little overhead of security, which may appear due to memory access permissions checks. That is a great advantage of FPGA based computation, maximising the gains.

None of the cited works claim any overhead to computation speed of FPGA for proposed security checks. However, they can introduce overheads on programming latency, as they require encryption or authentication of the bitstreams. Even without security checks re-programming an FPGA or a part of it is not fast. Thus, the FPGA manufacturers are working hard to improve their (partial) programming performance, so that hardware context-switch can be fast. Minimising the overhead of security checks over this context-switch can be achieved with cleverly picked crypto-algorithms, and protocols. Furthermore, the FPGA manufacturers can provide optimised ASIC crypto accelerators (similar to AES-NI instruction Intel CPUs provide) that base-bitstream can use to execute fast security checks on the accelerators.

6.9 Trusted and Confidential Computing

Trusted computing aims at the expected good behaviour of applications. It introduced *isolation* as an architectural protection of (part of) a software and its associated data with access control, *attestation* for verifying the integrity of it even at remote location, and *sealing* for making a set of resources available to a certain state of it. Various architectures have been designed to achieve these [46]. Intel SGX [49] could be considered as the most popular example, which places parts of applications within so-called enclaves for protecting them. It establishes a trusted computing base within the CPU package, considering any external component untrusted. However, making the FPGAs a part of the computation would require extending that trusted computing base. Unfortunately, so far no related work can achieve it to the best of our knowledge. However, Simon Johnson has expressed in his SysTEX keynote talk [39] that Intel's future plans will consider it. A basic solution would require sealing the enclaved applications and their hardware accelerators to each other. In addition, that should allow the enclaves to share (a part of) their the encrypted memory with the FPGA accelerators they work with. An elementary component for achieving that is to establish a secure communication channel between them, if-and-only-if they attest the trusted state to each other. That would require enhancing the attestation mechanisms on the FPGAs, supplying them with proficient key establishment solutions.

Trusted computing paves the way to confidential computing with the cross-industry efforts under the roof of Confidential Computing Consortium [11]. It aims at protecting the confidentiality of the processed data in the cloud. That is a crucial requirement especially for machine learning, and data analytics applications. For these applications, accelerated cloud platforms are interesting targets to provide the computation power; however, their data sets are highly valuable and need protection from untrusted cloud service providers. Therefore, FPGA accelerated computation should support the features of trusted computing so that a leverage could be obtained for confidential computing.

7 CONCLUSION

The incorporation of FPGAs into the cloud for domain specific acceleration introduces radical changes to the computer architectures, and creates concerns. The biggest concerns are a deficient trust model, the lack of intellectual property protection, and the secure shared use of hardware resources, amongst others. In this paper, we present an assessment of existing trust models, and corresponding security solutions to shed light on their shortcomings. We also surveyed the literature for projects offering possible solutions. In addition, we provide a discussion on the drawbacks of these work, and point out ways to improve them.

ACKNOWLEDGMENTS

This work was supported in part by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Centre "Invasive Computing" (SFB/TR 89), the KU Leuven Research Council through C16/15/058, and ERC Advanced Grant 695305.

REFERENCES

[1] 2013. *Xilinx Announces Full Production of its Entire Zynq-7000 All Programmable SoC Family.* https://www.xilinx.com/news/press/2013/xilinx-announces-full-production-of-its-entire-zynq-7000-all-programmable-soc-family.html

[2] 2015. IEEE Recommended Practice for Encryption and Management of Electronic Design Intellectual Property (IP). *IEEE Std 1735-2014 (Incorporates IEEE Std 1735-2014/Cor 1-2015)* (Sep. 2015). https://doi.org/10.1109/IEEESTD.2015.7274481

[3] 2015. *Intel Acquisition of Altera.* https://newsroom.intel.com/press-kits/intel-acquisition-of-altera/

[4] 2016. *Developer Preview - EC2 Instances (F1) with Programmable Hardware.* https://aws.amazon.com/blogs/aws/developer-preview-ec2-instances-f1-with-programmable-hardware/

[5] 2017. *Baidu Deploys Xilinx FPGAs in New Public Cloud Acceleration Services.* https://www.xilinx.com/news/press/2017/baidu-deploys-xilinx-fpgas-in-new-public-cloud-acceleration-services.html

[6] 2017. *Intel FPGAs Power Acceleration-as-a-Service for Alibaba Cloud.* https://newsroom.intel.com/news/intel-fpgas-power-acceleration-as-a-service-alibaba-cloud/

[7] 2017. *Xilinx Powers Huawei FPGA Accelerated Cloud Server.* https://www.xilinx.com/news/press/2017/xilinx-powers-huawei-fpga-accelerated-cloud-server.html

[8] 2017. *Xilinx Selected by Alibaba Cloud for Next-Gen FPGA Cloud Acceleration.* https://www.xilinx.com/news/press/2017/xilinx-selected-by-alibaba-cloud-for-next-gen-fpga-cloud-acceleration.html

[9] 2018. *Tencent Cloud - Instance Type FPGA FX2.* https://intl.cloud.tencent.com/document/product/213/11518#FX2

[10] 2018. *Twitch Chooses Xilinx to Enable its Broadcast-quality Livestream of eSports.* https://forums.xilinx.com/t5/Xilinx-Xclusive-Blog/Live-from-XDF-Twitch-Chooses-Xilinx-to-Enable-its-Broadcast/ba-p/894711

[11] 2019. *Confidential Computing Consortium.* https://confidentialcomputing.io/

[12] 2019. *Open Programmable Acceleration Engine.* https://opae.github.io/1.0.2/index.html.

[13] 2019. *What are field-programmable gate arrays (FPGA) and how to deploy.* https://docs.microsoft.com/en-us/azure/machine-learning/service/how-to-deploy-fpga-web-service

[14] François Abel, Jagath Weerasinghe, Christoph Hagleitner, Beat Weiss, and Stephan Paredes. 2017. An FPGA Platform for Hyperscalers. In *25th IEEE Annual Symposium on High-Performance Interconnects, HOTI 2017, Santa Clara, CA, USA, August 28-30, 2017.* IEEE Computer Society, 29–32. https://doi.org/10.1109/HOTI.2017.13

[15] ARM 2013. *AMBA AXI and ACE Protocol Specification.* ARM. https://developer.arm.com/documentation/ihi0022/e/

[16] Mikhail Asiatici, Nithin George, Kizheppatt Vipin, Suhaib A. Fahmy, and Paolo Ienne. 2017. Virtualized Execution Runtime for FPGA Accelerators in the Cloud. *IEEE Access* 5 (2017), 1900–1910. https://doi.org/10.1109/ACCESS.2017.2661582

[17] Arnab Bag, Sikhar Patranabis, Debapriya Basu Roy, and Debdeep Mukhopadhyay. 2018. Cryptographically Secure Multi-Tenant Provisioning of FPGAs. *CoRR* abs/1802.04136 (2018). arXiv:1802.04136 http://arxiv.org/abs/1802.04136

[18] Ernie Brickell and Jiangtao Li. 2007. Enhanced privacy id: a direct anonymous attestation scheme with enhanced revocation capabilities. In *Proceedings of the 2007 ACM Workshop on Privacy in the Electronic Society, WPES 2007, Alexandria, VA, USA, October 29, 2007.* 21–30. https://doi.org/10.1145/1314333.1314337

[19] Ernest F. Brickell, Jan Camenisch, and Liqun Chen. 2004. Direct anonymous attestation. In *Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS 2004, Washington, DC, USA, October 25-29, 2004.* 132–145. https://doi.org/10.1145/1030083.1030103

[20] Gaetan Canivet, Paolo Maistri, Régis Leveugle, Jessy Clédière, Florent Valette, and Marc Renaudin. 2011. Glitch and Laser Fault Attacks onto a Secure AES Implementation on a SRAM-Based FPGA. *J. Cryptology* 24, 2 (2011), 247–268. https://doi.org/10.1007/s00145-010-9083-9

[21] Adrian M. Caulfield, Eric S. Chung, Andrew Putnam, Hari Angepat, Jeremy Fowers, Michael Haselman, Stephen Heil, Matt Humphrey, Puneet Kaur, Joo-Young Kim, Daniel Lo, Todd Massengill, Kalin Ovtcharov, Michael Papamichael, Lisa Woods, Sitaram Lanka, Derek Chiou, and Doug Burger. 2016. A cloud-scale acceleration architecture. In *49th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2016, Taipei, Taiwan, October 15-19, 2016.* 7:1–7:13. https://doi.org/10.1109/MICRO.2016.7783710

[22] Fei Chen, Yi Shan, Yu Zhang, Yu Wang, Hubertus Franke, Xiaotao Chang, and Kun Wang. 2014. Enabling FPGAs in the cloud. In *Computing Frontiers Conference, CF'14, Cagliari, Italy - May 20 - 22, 2014.* 3:1–3:10. https://doi.org/10.1145/2597917.2597929

[23] Animesh Chhotaray, Adib Nahiyan, Thomas Shrimpton, Domenic Forte, and Mark Mohammad Tehranipoor. 2017. Standardizing Bad Cryptographic Practice: A Teardown of the IEEE Standard for Protecting Electronic-design Intellectual Property. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu (Eds.). ACM, 1533–1546. https://doi.org/10.1145/3133956.3134040

[24] Eric Chung and Jeremy Fowers. 2017. *Accelerating Persistent Neural Networks at Datacenter Scale.* https://www.hotchips.org/archives/2010s/hc29/

[25] Florian Devic, Lionel Torres, and Benoît Badrignans. 2010. Secure Protocol Implementation for Remote Bitstream Update Preventing Replay Attacks on FPGA. In *International Conference on Field Programmable Logic and Applications, FPL 2010, August 31 2010 - September 2, 2010, Milano, Italy.* 179–182. https://doi.org/10.1109/FPL.2010.44

[26] Florian Devic, Lionel Torres, Jérémie Crenne, Benoît Badrignans, and Pascal Benoit. 2012. SecURe DPR: Secure update preventing replay attacks for dynamic partial reconfiguration. In *22nd International Conference on Field Programmable Logic and Applications (FPL), Oslo, Norway, August 29-31, 2012*. 57–62. https://doi.org/10.1109/FPL.2012.6339241

[27] Saar Drimer, Tim Güneysu, Markus G Kuhn, and Christof Paar. 2008. Protecting multiple cores in a single FPGA design. *Draft available at http://www.cl.cam.ac.uk/sd410/, written May* (2008).

[28] Saar Drimer and Markus G. Kuhn. 2009. A Protocol for Secure Remote Updates of FPGA Configurations. In *Reconfigurable Computing: Architectures, Tools and Applications, 5th International Workshop, ARC 2009, Karlsruhe, Germany, March 16-18, 2009. Proceedings*. 50–61. https://doi.org/10.1007/978-3-642-00641-8_8

[29] Ken Eguro and Ramarathnam Venkatesan. 2012. FPGAs for trusted cloud computing. In *22nd International Conference on Field Programmable Logic and Applications (FPL), Oslo, Norway, August 29-31, 2012*. 63–70. https://doi.org/10.1109/FPL.2012.6339242

[30] Karim Eldefrawy, Gene Tsudik, Aurélien Francillon, and Daniele Perito. 2012. SMART: Secure and Minimal Architecture for (Establishing Dynamic) Root of Trust. In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*.

[31] Ilias Giechaskiel, Kasper Bonne Rasmussen, and Ken Eguro. 2018. Leaky Wires: Information Leakage and Covert Communication Between FPGA Long Wires. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security, AsiaCCS 2018, Incheon, Republic of Korea, June 04-08, 2018*. 15–27. https://doi.org/10.1145/3196494.3196518

[32] Tim Güneysu, Bodo Möller, and Christof Paar. 2007. Dynamic Intellectual Property Protection for Reconfigurable Devices. In *2007 International Conference on Field-Programmable Technology, ICFPT 2007, Kitakyushu, Japan, December 12-14, 2007*. 169–176. https://doi.org/10.1109/FPT.2007.4439246

[33] Boeui Hong, Han-Yee Kim, Minsu Kim, Taeweon Suh, Lei Xu, and Weidong Shi. 2018. FASTEN: An FPGA-Based Secure System for Big Data Processing. *IEEE Design & Test* 35, 1 (2018), 30–38. https://doi.org/10.1109/MDAT.2017.2741464

[34] Ted Huffmire, Cynthia Irvine, Thuy D Nguyen, Timothy Levin, Ryan Kastner, and Timothy Sherwood. 2010. *Handbook of FPGA design security*. Springer Science & Business Media.

[35] Intel 2020. *Intel Programmable Acceleration Card (PAC) with Intel Arria 10 GX FPGA Datasheet*. Intel. https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ds/ds-pac-a10.pdf

[36] Aws Ismail and Lesley Shannon. 2011. FUSE: Front-end user framework for O/S abstraction of hardware accelerators. In *2011 IEEE 19th Annual International Symposium on Field-Programmable Custom Computing Machines*. IEEE, 170–177.

[37] Matthew Jacobsen, Dustin Richmond, Matthew Hogains, and Ryan Kastner. 2015. RIFFA 2.1: A Reusable Integration Framework for FPGA Accelerators. *TRETS* 8, 4 (2015), 22:1–22:23. https://doi.org/10.1145/2815631

[38] Chenglu Jin, Vasudev Gohil, Ramesh Karri, and Jeyavijayan Rajendran. 2020. Security of Cloud FPGAs: A Survey. *arXiv preprint arXiv:2005.04867* (2020).

[39] Simon Johnson. 2019. *Systex 2019 Keynote Talk - Scaling Towards Confidential Computing*. https://systex.ibr.cs.tu-bs.de/systex19/slides/systex19-keynote-simon.pdf

[40] Krzysztof Kepa, Fearghal Morgan, Krzysztof Kosciuszkiewicz, and Tomasz Surmacz. 2008. SeReCon: A Secure Dynamic Partial Reconfiguration Controller. In *IEEE Computer Society Annual Symposium on VLSI, ISVLSI 2008, 7-9 April 2008, Montpellier, France*. 292–297. https://doi.org/10.1109/ISVLSI.2008.61

[41] Nadir Khan, Arthur Silitonga, Brian Pachideh, Sven Nitzsche, and Jürgen Becker. 2019. Secure Local Configuration of Intellectual Property Without a Trusted Third Party. In *Applied Reconfigurable Computing - 15th International Symposium, ARC 2019, Darmstadt, Germany, April 9-11, 2019, Proceedings*. 137–146. https://doi.org/10.1007/978-3-030-17227-5_11

[42] Oliver Knodel, Paul R Genssler, Fredo Erxleben, and Rainer G Spallek. 2018. FPGAs and the Cloud–An Endless Tale of Virtualization, Elasticity and Efficiency. *International Journal on Advances in Systems and Measurement* 11, 3 (2018).

[43] Jonas Krautter, Dennis R. E. Gnad, and Mehdi Baradaran Tahoori. 2018. FPGAhammer: Remote Voltage Fault Attacks on Shared FPGAs, suitable for DFA on AES. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2018, 3 (2018), 44–68. https://doi.org/10.13154/tches.v2018.i3.44-68

[44] 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 Trans. Cryptogr. Hardw. Embed. Syst.* 2018, 3 (2018), 573–595. https://doi.org/10.13154/tches.v2018.i3.573-595

[45] Enno Lübbers and Marco Platzner. 2009. ReconOS: Multithreaded programming for reconfigurable computers. *ACM Trans. Embedded Comput. Syst.* 9, 1 (2009), 8:1–8:33. https://doi.org/10.1145/1596532.1596540

[46] Pieter Maene, Johannes Götzfried, Ruan de Clercq, Tilo Müller, Felix C. Freiling, and Ingrid Verbauwhede. 2018. Hardware-Based Trusted Computing Architectures for Isolation and Attestation. *IEEE Trans. Computers* 67, 3 (2018), 361–374. https://doi.org/10.1109/TC.2017.2647955

[47] Roel Maes, Dries Schellekens, and Ingrid Verbauwhede. 2012. A Pay-per-Use Licensing Scheme for Hardware IP Cores in Recent SRAM-Based FPGAs. *IEEE Trans. Information Forensics and Security* 7, 1 (2012), 98–108. https://doi.org/10.1109/TIFS.2011.2169667

[48] Kaspar Matas, Tuan La, Nikola Grunchevski, Khoa Pham, and Dirk Koch. 2020. Invited Tutorial: FPGA Hardware Security for Datacenters and Beyond. In *The 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. 11–20.

[49] Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V. Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R. Savagaonkar. 2013. Innovative instructions and software model for isolated execution. In *HASP 2013, The Second Workshop on Hardware and Architectural Support for Security and Privacy, Tel-Aviv, Israel, June 23-24, 2013*. 10. https://doi.org/10.1145/2487726.2488368

[50] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. 1996. *Handbook of Applied Cryptography*. CRC Press. http://cacr.uwaterloo.ca/hac/

[51] Andrew Putnam, Adrian M. Caulfield, Eric S. Chung, Derek Chiou, Kypros Constantinides, John Demme, Hadi Esmaeilzadeh, Jeremy Fowers, Gopi Prashanth Gopal, Jan Gray, Michael Haselman, Scott Hauck, Stephen Heil, Amir Hormati, Joo-Young Kim, Sitaram Lanka, James R. Larus, Eric Peterson, Simon Pope, Aaron Smith, Jason Thong, Phillip Yi Xiao, and Doug Burger. 2014. A reconfigurable fabric for accelerating large-scale datacenter services. In *ACM/IEEE 41st International Symposium on Computer Architecture, ISCA 2014, Minneapolis, MN, USA, June 14-18, 2014*. 13–24. https://doi.org/10.1109/ISCA.2014.6853195

[52] Chethan Ramesh, Shivukumar B. Patil, Siva Nishok Dhanuskodi, George Provelengios, Sébastien Pillement, Daniel Holcomb, and Russell Tessier. 2018. FPGA Side Channel Attacks without Physical Access. In *26th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM 2018, Boulder, CO, USA, April 29 - May 1, 2018*. 45–52. https://doi.org/10.1109/FCCM.2018.00016

[53] Kurt Rosenfeld and Ramesh Karri. 2010. Attacks and Defenses for JTAG. *IEEE Design & Test of Computers* 27, 1 (2010), 36–47. https://doi.org/10.1109/MDT.2010.9

[54] Sujoy Sinha Roy, Furkan Turan, Kimmo Järvinen, Frederik Vercauteren, and Ingrid Verbauwhede. 2019. FPGA-Based High-Performance Parallel Architecture for Homomorphic Computing on Encrypted Data. In *25th IEEE International Symposium on High Performance Computer Architecture, HPCA 2019, Washington, DC, USA, February 16-20, 2019*. 387–398. https://doi.org/10.1109/HPCA.2019.00052

[55] Hayden Kwok-Hay So and Robert W. Brodersen. 2008. A unified hardware/software runtime environment for FPGA-based reconfigurable computers using BORPH. *ACM Trans. Embedded Comput. Syst.* 7, 2 (2008), 14:1–14:28. https://doi.org/10.1145/1331331.1331338

[56] Pawel Swierczynski. 2018. *Bitstream-based attacks against reconfigurable hardware*. Ph.D. Dissertation. Ruhr University Bochum, Germany. https://hss-opus.ub.ruhr-uni-bochum.de/opus4/frontdoor/index/index/docId/6063

[57] Pawel Swierczynski, Marc Fyrbiak, Philipp Koppe, Amir Moradi, and Christof Paar. 2017. Interdiction in practice - Hardware Trojan against a high-security USB flash drive. *J. Cryptographic Engineering* 7, 3 (2017), 199–211. https://doi.org/10.1007/s13389-016-0132-7

[58] SymbiFlow2019. 2018. *SymbiFlow - open source FPGA tooling for rapid innivation*. https://symbiflow.github.io/

[59] Shanquan Tian, Wenjie Xiong, Ilias Giechaskiel, Kasper Rasmussen, and Jakub Szefer. 2020. Fingerprinting Cloud FPGA Infrastructures. In *The 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. 58–64.

[60] Stephen M Trimberger and Jason J Moore. 2014. FPGA security: Motivations, features, and applications. *Proc. IEEE* 102, 8 (2014), 1248–1265.

[61] Furkan Turan and Ingrid Verbauwhede. 2020. Proxy Re-Encryption for Accelerator Confidentiality in FPGA-Accelerated Cloud. Cryptology ePrint Archive, Report 2020/805. https://eprint.iacr.org/2020/805.

[62] Anuj Vaishnav, Khoa Dang Pham, and Dirk Koch. 2018. A Survey on FPGA Virtualization. In *28th International Conference on Field Programmable Logic and Applications, FPL 2018, Dublin, Ireland, August 27-31, 2018*. 131–138. https://doi.org/10.1109/FPL.2018.00031

[63] Jo Vliegen, Nele Mentens, and Ingrid Verbauwhede. 2013. A single-chip solution for the secure remote configuration of FPGAs using bitstream compression. In *2012 International Conference on Reconfigurable Computing and FPGAs, ReConFig 2013, Cancun, Mexico, December 9-11, 2013*. 1–6. https://doi.org/10.1109/ReConFig.2013.6732330

[64] Jo Vliegen, Md Masoom Rabbani, Mauro Conti, and Nele Mentens. 2019. SACHa: Self-Attestation of Configurable Hardware. In *Design, Automation & Test in Europe Conference & Exhibition, DATE 2019, Florence, Italy, March 25-29, 2019*. 746–751. https://doi.org/10.23919/DATE.2019.8714775

[65] Herbert Walder and Marco Platzner. 2004. A Runtime Environment for Reconfigurable Hardware Operating Systems. In *Field Programmable Logic and Application, 14th International Conference , FPL 2004, Leuven, Belgium, August 30-September 1, 2004, Proceedings*. 831–835. https://doi.org/10.1007/978-3-540-30117-2_84

[66] Qian Wang, Cong Wang, Jin Li, Kui Ren, and Wenjing Lou. 2009. Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing. In *Computer Security - ESORICS 2009, 14th European Symposium on Research in Computer Security, Saint-Malo, France, September 21-23, 2009. Proceedings*. 355–370. https://doi.org/10.1007/978-3-642-04444-1_22

[67] Zane Weissman, Thore Tiemann, Daniel Moghimi, Evan Custodio, Thomas Eisenbarth, and Berk Sunar. 2019. Jack-Hammer: Efficient Rowhammer on Heterogeneous FPGA-CPU Platforms. arXiv:1912.11523 [cs.CR]

[68] Xilinx [n.d.]. *XAPP1283 - Internal Programming of BBRAM and eFUSEs*. Xilinx. https://www.xilinx.com/support/documentation/application_notes/xapp1283-internalprogramming-bbram-efuses.pdf 1.1.

[69] Jiansong Zhang, Yongqiang Xiong, Ningyi Xu, Ran Shu, Bojie Li, Peng Cheng, Guo Chen, and Thomas Moscibroda. 2017. The Feniks FPGA Operating System for Cloud Computing. In *Proceedings of the 8th Asia-Pacific Workshop on Systems, Mumbai, India, September 2, 2017*. 22:1–22:7. https://doi.org/10.1145/3124680.3124743

[70] Li Zhang and Chip-Hong Chang. 2014. A Pragmatic Per-Device Licensing Scheme for Hardware IP Cores on SRAM-Based FPGAs. *IEEE Trans. Information Forensics and Security* 9, 11 (2014), 1893–1905. https://doi.org/10.1109/TIFS.2014.2355043

[71] Mark Zhao and G. Edward Suh. 2018. FPGA-Based Remote Power Side-Channel Attacks. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*. 229–244. https://doi.org/10.1109/SP.2018.00049

Received March 2020