A Survey on Security Features in Modern FPGAs

R. Druyer, L. Torres, P. Benoit
LIRMM, UMR 5506, Université de Montpellier,
161 rue Ada, Montpellier, France

P.V. Bonzom, P. Le-Quere
ATOS SAS
68 rue Jean Jaurès, Les Clayes-Sous-Bois, France

Abstract— **Security is a major challenge for the design on FPGAs. This one applies to different levels: IP protection, information confidentiality and denial of service. To assist the application designer in this purpose, FPGA vendors provide dedicated features, which address potential security breaches of their devices. In this paper, after setting up a complete FPGA threat model, we compare relevant functionalities of the most advanced products of Altera, Microsemi and Xilinx. The goal of this paper is to evaluate whether the security features embedded on the current FPGAs address the threat model.**

Keywords—FPGA, Security, Application, Altera, Microsemi, Xilinx

I. INTRODUCTION

FPGA market keeps growing years after years and these circuits stay an attractive solution not only for ASIC prototyping but also for application fields that take benefits of its advantages compared to ASIC. Briefly, its reasonable cost for a low- and mid-range volume, and its reconfiguration capability allows an easier debugging process. Automotive, avionics or military are critical application fields in which the FPGA has a large place and where the security issue is of a paramount importance. FPGA vendors try to address security threats by providing a set of features protecting the device and the user application availability, confidentiality and integrity. In the first section of this paper, we introduce a model including all the main security threats that may be encountered by application designers when working on FPGA. Then we draw up a table summarizing the security features currently provided by the FPGA vendors: Altera, Microsemi and Xilinx. Finally, we analyze the adequacy of these security features according to the threat model.

II. FPGA THREAT MODEL

The motivations of an attacker targeting FPGAs are just a few. He may desire to clone or analyse the configuration loaded into the FPGA, which is called intellectual property (IP) theft. Or, he may want to compromise the integrity or confidentiality of data processed by the running application.

A. Intellectual Property Theft

A first motivation is the IP theft. Concerning the industrial aspect, vendors protect the internal hardware architecture of their chips to avoid that a third-party can steal their IPs. But in the case of a FPGA application designer, to secure its intellectual property, his goal is to ensure that the bitstream configuration file generated to program the FPGA with the application is protected. An application designer is exposed to two types of IP theft threats: cloning and reverse engineering and both require the bitstream.

This configuration file contains the value of each programmed cell of the targeted chip. As each FPGA is different in terms of programmable resources, a given bitstream is generally compatible with a single device type. The bitstream file is unintelligible for someone without the knowledge of the hardware implementation of the programmable cells. However to achieve cloning only the plaintext bitstream extraction is needed to configure any FPGA devices with the same characteristics.

If an attacker wants to go further, analyses and modifies the application, he has to perform reverse engineering of the bitstream [22]. His goal is to rebuild the netlist (file describing how the logic elements within the device are connected with each other) or the Register Transfer Level (RTL) file. The exploitation of one of these files, allows extracting specific design parts, to improve the application and to use it for other circuits, whatever the technology (FPGAs or ASICs). However, bitstream format are now kept confidential by the vendors [6] [11], [3], coupled with an average design size and complexity increase, this makes the bitstream reversing a laborious and challenging task. However, this is not considered as impossible. Furthermore, attacks on the software development kits or FPGA hardware can be used to facilitate the intellectual property theft [19].

1) FPGA Technology

The technology has a large impact on the device security and performances, it refers to the memory type used for the configuration cells in the FPGA. Since there is no perfect choice, we will present the characteristics and specificities of each technology provided by the three vendors.

SRAM-FPGA is by far the most used and sold FPGA type. It takes benefits of the popularity of CMOS manufacturing process that is at least of 2 to 3 generation ahead of the others, providing a greater integration density, a better power efficiency and faster memory access times. However this memory is volatile, meaning that the device must be programed at each start-up. The bitstream must be stored in a non-volatile memory that typically located in an external chip and it is sent through the programming interface to the configuration cells, creating the opportunity for an attacker to probe the programming channel (*e.g.* JTAG) in order to extract the bitstream. Whether an attacker wishes to extract the value from the configuration cells within the FPGA using invasive attacks and reverse engineering, it is practically unfeasible due to the volatile nature of the SRAM [4]. Removing the package and the metal layers to access to the configuration cells will cause the FPGA configuration blanking. In one hand, since all FPGA technology types are resilient to invasive attacks, SRAM-based ones are considered the less secure due to the start-up reconfiguration requirement. In the other hand, SRAM-based FPGAs are the only choice possible for the most performance hungry applications.

The second type of FPGA technology is flash memory-based. It is an interesting choice in security application field thanks to its non-volatile nature. Thus, removing the start-up reconfiguration requirement and eliminating opportunities for an attacker to probe the bitstream. In terms of performances, flash memory is a low-power technology since no energy is required to maintain the cell value. Nevertheless, as we said before, the SRAM manufacturing process of the SRAM is more advanced and has better overall performances. Even if the flash memory is non-volatile, invasive reverse engineering is not an easy task. The flash cells are distributed all over the chip making the physical probes positioning difficult, as well as reconstructing a bitstream or a netlist with gathered values. Furthermore, micro-probing flash cell can destroy the charge on the floating gate and removes the contained value [2]. All these parameters make the configuration flash cells reverse engineering a very challenging, time consuming and costly task. In this case, an attack has no sense since it cost more than what you get in return. Be careful not to confuse flash-based FPGA with SRAM-based FPGA containing an internal flash memory. For example, the Spartan3AN [16] is a SRAM-based device including an internal flash able to store few bitstreams loadable at start-up. It is almost secure as flash-based FPGAs since the reconfiguration channel is not accessible from the outside of the chip. To conclude about flash FPGAs, when using external reconfiguration in order to update the application or fix some bugs, it is vulnerable to bitstream probing just as SRAM FPGAs.

Antifuse FPGA is also a technology claimed for its advantages in terms of security. The antifuse cells are programmed using a large current that definitively set a logic value. The device is programmed once and retains the configuration indefinitely, meaning that there is no need to reprogram the device at each power cycle. However, with this FPGA type reconfiguration becomes completely impossible, a device configured with flawed application cannot be updated. In terms of reverse engineering, this technology is very resistant to attacks. The electrical short circuit giving the value of the antifuse cannot be viewed from the top, thus to observe, it a cross-section of the antifuse is necessary [10] and state of a programmed antifuse is very hard to distinguish from an unprogrammed one [2]. Moreover a device counts millions antifuse and just a small percentage of it is programmed. The cost of a successful an attack on antifuse using current reverse engineering techniques must be astronomical.

To resume, antifuse is considered as the most secure technology due to its one-time programmable feature, however a flawed application cannot be updated, that can be critical if it contains security breaches. Concerning flash memory, the real benefit in terms of security is its non-volatile nature but using reconfiguration may exposes a flash device to the same vulnerabilities as a SRAM one. For the latest, SRAM is a suitable choice for high-performance applications.

2) Bitstream Probing

The bitstream stored in an external memory can be easily intercepted with an electrical probe during its transfer to FPGA through the programming interface. An attacker having a physical access to a SRAM or flash FPGA during the reconfiguration process can implement this attack. The advantage of antifuse is whether the programming step takes place in a trusted environment, even if the FPGA is then employed in an untrusted location the bitstream is practically impossible to extract. SRAM seems more vulnerable to bitstream probing because typically, the bitstream is loaded when the device is turned on. Thus, it is easier for an attacker who controls the power supply to trigger a reconfiguration. Flash FPGA must meet more specific conditions before triggering a reconfiguration.

3) Bitstream Decryption Key Stealing

Actually the solution chosen by the FPGA vendors to protect the bitstream confidentiality is the encryption (details in section IV). This do not prevent the bitstream to be stolen but in the case where the attacker has only retrieved the encrypted bitstream, he will not be able to reverse it. Moreover, the cloning is impossible if both the attacked and the cloned FPGA do not share the same key. At this point, the bitstream security lies on the key confidentiality, the application designer needs the insurance that the key cannot be extracted by any means. In this section, we give on overview of the potential threats related to decryption key during the different design phases.

The first threat concerning the decryption key takes place during its loading to the FPGA. The attacker must not have physical access to the channel used to load the key to FPGA during this phase. If the application designer wants to update the key on the field it must be transferred in an encrypted form.

Microsemi set a secret factory key during the FPGA manufacturing and gives the factory keys database to the buyer. This database must be stored in a safe place because it may be easier for an attacker to target a poorly protected server to retrieve FPGA factory keys than attack the secure device directly. Within the FPGA, the internal key storage have to be protected from readback attacks. It consists of using a debugging interface to read the internal FPGA data. Most of the time, FPGA vendors claim to provide read-protected bitstream decryption key storage. But for instance some researchers [18] have found an undocumented command into the JTAG interface of the Microsemi high-level security FPGA. This command gave a complete access to a set of security features including the bitstream decryption key.

One of the most important threat occurs during the bitstream decryption. The circuit leakages (power consumption, EM, temperature or timing variations) which reflect the activity during the decryption are correlated with the input data to reconstruct the key value, it is called the side-channel analysis (SCA). Some successful side-channel attacks have be reported for Altera Stratix-II and Stratix-III [19], Microsemi ProASIC3 [20] and Xilinx Virtex-II [21] that used bitstream AES-128 encryption. These papers demonstrate that SCA techniques are incredibly efficient with an affordable cost. We can note that for the attacks against the Stratix-II and Stratix-III [19], weaknesses residing within the development software Quartus are exploited. By decompiling the Quartus DLL files, the researchers have found which key derivation technique is used for generating the real bitstream decryption key. Furthermore, they have identified the mode of operation of AES used for the bitstream decryption inside the device. This highlights the fact that the security of the development software has also a crucial importance and FPGA.

4) Bitstream Readback Attacks

Programming interfaces like the Join Test Action Group (JTAG) work as small microcontrollers, actually a set of commands can be sent to the interface to achieve different programming or debugging functions. Generally, one command allows to retrieve the bitstream from the FPGA. It is intended to verify the bitstream integrity to see whether it contains error once the device is programmed. Nevertheless, if this command has not been deactivated before the device is on the field, it is a wonderful opportunity for an attacker to easily extract the bitstream.

5) FPGA Genuineness

The risk of buying a counterfeited FPGA is non-zero. Old FPGAs relabeled and sold as new were found in US Navy aircraft [23]. A counterfeited FPGA may seem to work like an authentic one, but a hardware Trojan or a backdoor may be present into it and open any sort of security breaches. We can imagine that a compromised device can leak any kinds of data or create an internal shortcut that causes the chip destruction.
But even if a vendor can guarantee the authenticity of the device being bought, nowadays the issue concerning the device confidence is of a crucial importance. Microsemi claims to have the most secure devices on the market and possesses a certain numbers of certificate (*e.g.* coming from the U.S. Department-of-Defense [1]). Their FPGAs are particularly used in military and aerospace critical applications. But the revelation of a probable backdoor in the JTAG interface of Microsemi ProASIC3 FPGA [18] illustrates the fact that even with certifications, the confidence granted to a device cannot be total. In fact, the better way for a company to be confident of a final product is to be proprietary of all the technology employed for it and to control all the flow from the design to the fabrication, but for most of them it is unconceivable. Starting from that, a company can increase its confidence in a device thanks to certifications coming from its government and features implemented by the vendor that aim to make a device the most secure possible.

B. User Application Data Compromise

The second part of our threat model gathers the attacks used to steal confidential information or modify user application performances. We gathered these two different goals in one section because most of the time attacks can be used to complete both purposes. Concerning denial of service (DOS) attacks, it is nonsense to study complex threats requiring a physical access to the circuit since a FPGA chip is far from indestructible. Nevertheless it is interesting to study how an adversary may attempt to deactivate a circuit using a distant access.

If an attacker is able to modify the FPGA configuration, it can reach different goals following its knowledge about the application and its mastery of the employed attack. Whether it can modify random configuration bits, we can imagine that some parts of the final application will be defective. In the case where of the modifications are located in critical application parts or their number is too important, the application will completely crash. Now if we assume that an adversary can precisely locate which bits he must target and he is able to modify their value, he can expect to extract secret information from the chip. The attacks targeting the FPGA configuration can take place during three separate phases described below.

1) Before the Configuration

For the sake of clarity, the post-configuration phase includes all the attacks occurring during the FPGA manufacturing process or user application design. Threats during these phases are mainly related to Trojan insertion. Firstly concerning this type of attacks on the FPGA architecture itself, if the integrated circuit is counterfeit it may contain a malicious part. In the case where the device is genuine, it may have a backdoor that is known by the vendor or not [18]. Secondly, a Trojan can be inserted into the user application. The increasing systems complexity makes the design verification process a costly task and the fast time to market encourages designers to shorten this phase. Third-party IPs became common, thus the application can contain a small code giving a privileged access to the system, leaking secret data or generating malfunctions.

2) During the Configuration

The configuration phase takes into account all the attacks that occurs while the bitstream is loaded into the FPGA. The bitstream snooping is an important threat. It can be used for IP theft purpose as we seen in previously, but not only. Once the bitstream is reversed to a netlist or high-level description (HDL) files, it can contains exploitable secret information. Moreover, it gives a strong knowledge on the application contents and its mapping into the FPGA and allows to perform additional attacks. FPGA may be part of a greater system and its hijacking may give access to even important data for secrecy extraction or denial of service. Considering the configuration phase, in some cases it is preferable that an attacker cannot load its own bitstream into the device, particularly if the FPGA is included into global system. Another threat is the back tracking. It consists of loading an old bitstream version of the same application, including security breaches that have since been fixed. This kind of attacks is effective to pass through authentication process whether the authentication code has not been modified after an important application update.

The bitstream integrity can be compromised during its transmission to the FPGA. If an attacker is able to modify the value of the bits transmitted to the FPGA, he can empirically observe the effects on the application and in the case where he has enough knowledge on the bitstream format or its contents, he will be able to deactivate important security functionality, disturb the application or extract secret information. Bitstreams have a particular format following the FPGA device characteristics and the details are not anymore public. Generally, a bitstream file has a header that contains programming information. When an attacker modifies an encrypted bitstream, he cannot know which bits will be affected after the decryption. Thus the bitstream format may become invalid and cannot be programmed into the device.

3) *After the Configuration*

Once the FPGA has been configured with the user application, hardware Trojan contained in the application or in the physical circuit architecture can be triggered to achieve malicious intents. Briefly, we can imagine that the malicious system part waits a certain time amount or a specific data sequence before it triggers. Then the Trojan can force the application to crash or the FPGA to burn with a shortcut. It can also send data to a specific interface in order to leak secret information. It can also modify random, specific configuration or application data to generate malfunctions. The Trojan can also be a backdoor that gives a privileged access to the system [18].

Fault attacks are another important threats that take place after the configuration. Fault sources are multiple. It can be fault injection from industrial laser or an antenna emitting electromagnetic (EM) waves targeting specific circuit parts. External condition like temperature, supply voltage or the clock source can be pushed out of their ranges to provoke multiple kinds of unexpected behaviors. Mainly, the goals of all these fault attacks are to change the value of memory cells (*e.g.* bit-flip or SEU/MEU). Since most of them necessitate a physical access, we assume that their main goal is to extract secret information from the target device or modify its behavior to a specific one. If the goal is just to destruct the circuit, there is much simpler way to do it.

The last threat that we see in this paper is the side-channel attacks. We have seen that they are mainly used against cryptographic operation to extract the secret key (*e.g.* during the bitstream decryption). Since a FPGA vendor ignores what kind of applications will be programmed into their device (cryptographic or not), it is hard to propose countermeasures against this kind of attacks.

III. Security features analysis

All the security features presented in this paper, including the TABLE I. were gathered using the latest datasheets and information that we could find on each vendor website. We only introduce the mechanisms provided by each company and this does not mean that a user cannot integrate its own security mechanisms. Since there are numerous security features and some are very complex, we have done our best to give a correct and precise summary of it. However you are encouraged to consult the original datasheets if you want more information about the security mechanisms and to ensure whether no mistake has been made in this paper.

A. *IP Protection*

1) *FPGA Technology*

The advantages of each technology have already been detailed in the section II, thus in this section we just detailed what type of FPGAs are provided by each vendor.

Xilinx and Altera lead the market with SRAM FPGAs and do not offer other alternative. For their devices the flash memory is only used as bitstream storage and typically as an external memory. But once, Xilinx have proposed a FPGA with a different architecture, the Spartan-3AN. It comprises an internal flash memory that can contain few bitstreams. This particularity makes the Spartan-3AN secure against bitstream probing because the configuration file storage is integrated within the FPGA circuit. Concerning the bitstream decryption key memory, both vendors have the same approach by providing two solutions. One-time programmable fuses or battery-backed RAM (BBRAM). In one hand, the fuses are quite resistant against reverse engineering, but since the decryption key count only 256 bit, the reverse engineering is more feasible than for the 50 million of configuration antifuses that count the IGLOO2. The fuses make the key snooping impossible since the value is kept permanently and not modified. In the other hand, BBRAM requires a constant energy supply from a battery to keep the key value. And the key loading must be done in a trusted place to avoid probing. The main advantage is that the key value is not permanent and can be modified. The key can be cleared by removing the power supply and used as a countermeasure. Microsemi chooses for its IGLOO2 a 65nm flash-based technology that is oriented for low-power application. A range of antifuse FPGA is also available but they do not embed as many security features as the IGLOO2.

TABLE I. Security features provided by FPGA Vendors (for their higher security level FPGA model)

	Altera (Cyclone III LS)	**Microsemi (IGLOO2)**	**Xilinx (7-series)**
.bit encryption/decryption	AES-256	AES-256 (anti-DPA)	AES-256
.bit authentication/integrity	Authentication using an external memory device.	Tag based on SHA-256	HMAC-SHA-256
.bit backtracking prevention	No	Yes (with versioning)	No
.bit readback deactivation	Yes (readback not supported by the device).	Yes	Yes (hardened triple-redundant protection logic).
Programming interface monitoring and disabling	JTAG command restriction (only with MAX-II CPLD)	JTAG monitoring and disabling. Disabling only for SPI external flash interface.	JTAG monitoring and disabling.
Configuration memory integrity checking	Continuous CRC-32.	Exportable keyed digest (Certif.-of-Conformance).	Continuous CRC.
.bit key encrypted loading	No	Yes	No
.bit key storage memory type	Battery-backed RAM (BBRAM) or eFUSE (key stored in a scrambled form).	Flash	BBRAM or eFUSE.
.bit key zeroization	Yes	Yes	Yes (BBRAM)
.bit secret factory decryption key	None	Yes	None
FPGA memory type	SRAM (60nm)	Flash (65nm)	SRAM (28nm)
Supply chain assurance	None	Device ID and X.509 device certificate.	57-bit device ID set in fuses. 32-bit eFUSE user dedicated.
Internal integrity tests	Continuous CRC-32 on the embedded RAM (only with MAX-II CPLD).	SHA-256 on the ROM (configuration fabric, security settings, keys and declared NVM pages). SECDED on eSRAM, eNVM and DDR controller).	Only on configuration memory.
External condition monitoring	Internal oscillator watchdog (only with MAX-II CPLD). Temperature sensor (on Arria V and Stratix IV, V).	Dual internal clocks monitoring, active metal mesh.	Temperature, voltage and internal clock monitors.
Active countermeasures	Key zeroization. Registers zeroization (configuration memory, user memory).	Device lockdown, complete device zeroization (with post verification), I/Os set to 'Z' state	Configuration memory zeroization, user flip-flop reset and outputs set to 'Z' state.
Hardware accelerators and security services	None	AES-128/256 (ECB, OFB, CRT, CBC), SHA-256, HMAC, ECC (anti-DPA), Keytree derivation algorithm (anti-DPA), Quiddikey™ PUF key storage, AMBA bus hardware firewalls.	None
Isolation/partitioning tool	Design Separation Flow	No	Isolation Design Flow

2) *Bitstream Encryption*

The countermeasure employed against bitstream probing is the encryption. Using a symmetric algorithm like the AES to cipher the configuration file sent to the FPGA makes the programming interface probing ineffective, since the extracted data are unintelligible. Currently the bitstream encryption algorithm proposed for the most secure devices of the three FPGA vendors studied is the same, namely the AES-256. In addition to that, Xilinx and Microsemi devices require a complete device configuration erasure before loading a new configuration. This is intended to prevent that a previously loaded bitstream can compromise the following one.

3) *Bitstream Decryption Key Protection*

When bitstream encryption is used, the cryptographic key protection is of a paramount importance. The AES algorithm when used with a sufficient key length has been proved resistant to brute-force attacks achieved with current computing resource (available for the common people). This is the reason why the key confidentiality is an important concern.

For the moment Altera and Xilinx devices only propose a single 256-bit loaded user-programmed key for bitstream decryption and does not offer encryption for it [4][13]. Concerning key storage, for most of the devices two choices are available: volatile battery-backed RAM (BBRAM) or non-volatile embedded fuses (eFUSEs), note that Altera Cyclone III LS only provides BBRAM. The BBRAM offers key updating possibility. And the power removing can be used to clear the key as a tamper event response. However, this technique requires a short power disconnection to clear the RAM content. The eFUSEs ensure that the data is kept even without power. It is a practical advantage in terms of maintenance but even if it is a challenging task, an attacker has all the time he needs to conduct a physical attack. Moreover a device keeps the same key value during all its lifetime. Using Cyclone III LS, the key cannot be read out through any interfaces, and the key is scrambled for storage. In this case key scrambling can just delay an attack, but it is not considered as a strong security mechanism. To ensure the key confidentiality during the transmission to the chip, in function of the Altera FPGA device, the user must supply one or two keys to the programming software [19]. The software applies a key derivation function giving the real key used for the bitstream encryption. The one or two keys given by the user are sent to the FPGA using the same derivation function as the software. This technique is not yet flawless, since researchers were able to found the key derivation function that was used for the Stratix-II and Stratix-III by reversing the programing software Quartus II [19].

An interesting point about Xilinx storage is that any read or write access to the key BBRAM causes its content to be cleared with the entire FPGA configuration.

Typically, when encryption is not used, key loading shall never be done in an untrusted location. As the best of our knowledge, Microsemi is the only FPGA vendor to provide an encryption mechanism for user keys loading; moreover it is mandatory for the IGLOO2 [8]. The Key-Loading Key (KLK) is used by default for loading user keys. Since this key is similar for a large number of devices it is not recommended to use it. Furthermore, the KLK is not usable for bitstream decryption. To load encrypted user keys or the encrypted bitstream it is preferable to use the Derived Factory Key (DFK). This key is automatically derived from the unique per device Factory Key (FK) loaded in Microsemi factory. The key derivation algorithm is anti-DPA patented. Another factory key provided by Microsemi is only integrated in larger devices. It is a random ECC private key associated with an Elliptic Curve Cryptography (ECC) hardware accelerator. The factory key (FK) and the user keys are symmetric (i.e. the same key is used for encryption and decryption), but this one is asymmetric (*e.g.* data encrypted with the public key can only be decrypted with the private key and vice versa). The public ECC key is automatically computed using the ECC engine, which is certified in the X.509 device certificate. This key pair can be used for establishing a shared symmetric key using Elliptic Curve Diffie-Hellman (ECDH) protocol for bitstream decryption or authentication. Microsemi supplies a unique database containing all the factory keys to the client. And all this factory keys can be used for loading the two user keys UEK1 & UEK2. User keys can serve for bitstream decryption or authentication and all the keys (except the KLK) can independently encrypt a bitstream part that configures only the FPGA fabric or the embedded Non-Volatile Memory (eNVM) contents. Concerning user keys storage into Microsemi FPGAs, a SRAM-PUF technology called Intrinsic-ID™ from Quiddikey is provided. This technology does not work like a classic memory. It does not store the key value "directly" but it uses a Physical Unclonable Function (PUF) to reconstruct the keys on the fly when powered. An enrolment phase is necessary to initialize the key value into the PUF engine. This technology is assumed to be resistant against invasive attacks since the key values are not directly contained in it. Furthermore, the PUF based on the SRAM process variations, makes the component unique for each device and not reproducible. For key verification, a challenge protocol is implemented and allows checking if a device possesses a specific key without exposing it. This mechanism can be deactivated in the FPGA security settings.

The key encryption makes the key loading process more secure in an untrusted location because a simple snooping attack become ineffective. With Microsemi devices, to ensure that decryption keys cannot be modified by anyone, a 256-bit passcode must be set during the first key injection process. This passcode is then mandatory to unlock the write access to the security settings, including the keys. The researcher, who has found the backdoor in the Microsemi ProASIC3, considers this mechanism strong against DPA attacks [18]. To be used in an untrusted location one-time-use passcodes should be implemented in future Microsemi development software version. Care must be taken using this feature because the security settings remain completely accessible until the device or the JTAG reset. An optional passcode only unlocking the second user key is also available. The weak point concerning the Microsemi FPGAs, is the discovery of a backdoor [18] in the JTAG of the ProASIC3. We do not have found recent information concerning backdoors in current Microsemi devices.

In the section II, we saw that DPA attacks were successfully performed against the AES-128 decryption key of the previous generation FPGA from each vendor. Microsemi is for the moment the one and the only FPGA Company to integrate DPA patented countermeasures from CRI for bitstream decryption. However, no successful DPA attack has been reported in the literature for the last FPGA generation: V and 10-series for Altera, IGLOO2 for Microsemi and 6 and 7-series for Xilinx.

4) Readback deactivation

Readback is initially a verification mechanism used to read out the FPGA configuration and verify if it has not been corrupted during the programming or after. Removing this function makes the verification process harder but it avoids that an attacker can easily read the entire FPGA configuration. The Altera MAX-II CPLD can be used with the Cyclone III LS [3] [4], to restrict the JTAG commands set and makes the configuration readback impossible. To recover all the JTAG commands, a reset of the interface is necessary but it makes the MAX-II triggers the FPGA reset (configuration and volatile keys). In Microsemi devices, the debugging features can be deactivated using security settings called lock-bits. These lock-bits are configurable only if the right 256-bit passcode is sent to the device [8]. To verify the device configuration, a hash of the configuration fabric is generated and keyed with the bitstream encryption key and exported out of the FPGA to the user. It is called the "Certificate-of-Conformance" and it can be switched off using a specific lock-bit. When using bitstream encryption with Xilinx devices, the readback circuitry for all interfaces is automatically disabled. This mechanism is integrated in a hardened triple-redundant logic. All the vendors provide readback deactivation but only Microsemi offers an alternative verification mechanism.

5) Guarantee of FPGA Genuineness

This feature tends to prove the device genuineness. Altera does not seem to provide any numeric ID or signature in their FPGAs. Microsemi IGLOO2 contains a 128-bit serial number and a 256-bit user design ID. Furthermore it stores a X.509 device certificate containing the device serial number, date code, device number and secret factory keys which are bound and signed by Microsemi. This certificate ensures that the specifications of a given FPGA, match with the specifications provided by the vendor. Xilinx provides a unique 57-bit Device DNA for each 6-series and 7-series FPGAs [14]. It is implemented in one-time programmable (OTP) fuses, making the number not modifiable. The DNA is accessible internally by the design and externally via JTAG. A dedicated 32-bit user eFUSE is also accessible within the FPGA fabric. These user identification mechanisms can be used by the application developer as a strong bitstream anti-cloning countermeasures, linking each configuration file to specific devices.

B. User Application Data Protection

1) Before the Configuration

As we said before, application data-based attacks during this phase are mainly related to Trojan insertion. A malicious function like a backdoor or a logic bomb may be inserted into the device manufacturing process or during the user application design. We have seen in the previous section, how some vendors prove that a device is not counterfeit. If the device is genuine, it can contains a backdoor anyway, due to negligence or by choice [18]. Since you don't have a complete knowledge on the hardware you are forced to believe a vendor when he pretends that no such things exist. We can hope that researchers continue to work on this subject to improve the general hardware security level. About logic bomb, if it triggers when it detects a specific bit sequence, the encryption of the data transmitted on the different application communication medium may reduce the chance of triggering. In addition of being a very expensive security mechanism, no FPGA vendors propose embedded bus data encryption mechanisms. Microsemi proposes hardware firewalls to manage AMBA bus masters. Read and write right accesses to embedded volatile/non-volatile FPGA memory and some external memory controllers can be restricted. Note that security firewalls can be directly designed by the user.

2) During the Configuration

The bitstream encryption used to avoid bitstream theft protects also the design against reverse engineering. Backtracking attacks can be avoided by modifying the bitstream encryption keys at each major application update. To ensure the bitstream integrity during its loading to the FPGA, hash algorithms like the SHA-256 are efficient. They have been studied to resist against reversing, meaning that it is practically impossible to reconstruct an input bit sequence from its hash. Altera does not seem to provide any bitstream hashing mechanism. Since an encrypted bitstream that is modified during its transmission may be totally corrupted after the decryption, it is quite likely that the Altera FPGA refuses it for programming, thus protecting it against a malicious modification. Microsemi programming software generates a SHA-256-based hash of the configuration transmitted with the encrypted bitstream. Then the FPGA can verify bitstream integrity. A similar mechanism is used in Xilinx devices, except that the SHA-256 hash is keyed, giving a Hashed Message Authentication Code (HMAC). Typically the key is transmitted with the hash into the encrypted bitstream. For this reason, we don't see the advantages of using HMAC rather than a simple SHA-256. In FPGA-based design, to avoid that everyone can program the device with its own bitstream, an authentication procedure is necessary. This is provided with the bitstream encryption key, because it is assumed that only a trusted party has it.

3) After the Configuration

Concerning fault injection attacks, mechanisms are provided by vendors to detect and correct unexpected system behaviors. For Altera, the MAX-II CPLD can be used to monitor the Cyclone III LS internal oscillator to detect an interruption. MAX-II can be also configured to continuously compute a Cycle Redundancy Check (CRC) of the configuration RAM and spot unexpected value modifications. A temperature sensor is embedded in the Arria V, Stratix IV & V to detect out of range environmental conditions [5]. Microsemi IGLOO2 integrates a set of integrity tests, SHA-256 for the security settings and declared eNVM pages. Single Error Correction and Dual Error Detection (SECDED) is used for eSRAM, eNVM and DDR controller. The clock behavior is checked and an active metal mesh detects invasive attacks. Xilinx FPGAs integrate a CRC for the configuration memory. Clock, temperature and voltage are monitored.

Whenever a tamper event is detected, active countermeasures can be automatically triggered. All the vendors provide device zeroization. Microsemi FPGAs can be totally lock while conserving data as tamper evidences. Both Altera and Xilinx provide the possibility to physically isolate region inside the FPGA with unused logic fences, providing a strong insurance for fault-tolerant and reliable applications [6] [15].

For the moment, Microsemi is the only company to provide cryptographic hardware accelerators inside their FPGAs. Namely AES-128/256, SHA-256, HMAC, anti-DPA patented ECC engine and Keytree key derivation algorithm engine. These hardware accelerators are valuable for cryptographic applications. To protect the FPGA application during runtime, Microsemi provides security settings known as lock-bits which are unlockable using a 256-bit anti-DPA patented passcode. These lock-bits can disable read/write accesses of the majority of memory areas, hardware accelerators and programming interfaces. An irreversible lock-bits can turn the device to a one-time programmable one.

IV. Conclusion

FPGAs are used in critical systems and the value of the applications keeps growing causing increases of attacks targeting these devices last years. Now nearly all FPGA vendors propose high-level security devices. We have studied the security features provided by market leading companies Altera and Xilinx. And also Microsemi who claims to lead the market of secure FPGAs. To evaluate the relevance of the security solutions, we drawn up a large FPGA threat model including the most popular attacks. To resume our study, Altera FPGAs embed countermeasures against the most current attacks like bitstream snooping or readback, but we can notice the absence of real encryption mechanism for key loading making this process unsecure in untrusted location. The necessity of the MAX-II CPLD for advanced security features (*e.g.* clock monitoring or CRC) may be incompatible with some designs. In addition to all Altera's security mechanisms, Xilinx provides anti-counterfeiting device ID, voltage and temperature external monitors. Xilinx's offer is fairly complete for the moment but Microsemi has the largest FPGA security features offer. Most of them are unique for the industry like anti-DPA patented hardware accelerators, factory keys, X.509 device certificate and encrypted key loading. However, unlike the other two firms, physical isolation tools and voltage/temperature monitors are not provided.

References

[1] Altera – "Whitepaper: Protecting the FPGA Design From Common Threats", 2009.

[2] Altera – AN 593: "Anti-tamper Protection for Cyclone III LS Devices", 2009.

[3] Altera – AN512: "Using the Design Security Feature in Stratix III Devices", 2009.

[4] https://www.altera.com/products/fpga/features/stx-design-security.html

[5] Altera – UG01074: "Altera Temperature Sensor IP Core User Guide", 2014.

[6] Altera – AN567: "Quartus II Design Separation Flow", 2009.

[7] Altera – AN556: "Using the Design Security Features in Altera FPGAs", 2015.

[8] Microsemi – "IGLOO2 FPGA Security and Reliability User's Guide", 2014.

[9] Microsemi – "Security FAQs", 2015.

[10] Microsemi – "Secure Architecture in Microsemi FPGAs and SoC FPGAs-an Overview", 2013.

[11] Microsemi – "Specify and Program Security Settings and Keys with SmartFusion 2 and IGLOO2 FPGAs", 2013.

[12] http://www.microsemi.com/products/fpga-soc/radtolerant-fpgas/military-aerospace-certifications

[13] Xilinx - WP365: "Solving Today's Design Security Concerns ", 2012.

[14] Xilinx – XAPP1084: "Developing Tamper Resistant Designs with Xilinx Virtex-6 and 7 Series FPGAs", 2013.

[15] Xilinx - "Isolation Design Flow for Xilinx 7 Series FPGAs or Zynq-7000 AP SoCs (ISE Tools)", XAPP1086, 2015.

[16] http://www.xilinx.com/products/design_resources/config_sol/s3/config_s3e.htm

[17] S.M. Trimbergern J.J. Moore "FPGA Security:Motivations, Features, and Applications", Proceedings of the IEEE | Vol. 102, No. 8, pp 1248-1265, August 2014.

[18] S. Skorobogatov, C. Woods, "Breakthrough silicon scanning discovers backdoor in military chip", 2012.

[19] P. Swierczynski, C. Paar, "Physical Security Evaluation of the Bitstream Encryption Mechanism of Altera Stratix II and Stratix III FPGAs", ACM Transactions on Reconfigurable Technology and Systems (TRETS), volume 7 issue 4 article No. 34, January 2015.

[20] S. Skorobogatov, C. Woods, "In the blink of an eye: There goes your AES key", IACR Cryptology ePrint Archive 2012/296, 2012.

[21] A. Moradi, T. Kasper, "On the Vulnerability of FPGA Bitstream Encryption against Power Analysis Attacks", Cryptology ePrint Archive, Report 2011/390, 2011.

[22] J-B. Note, E. Rannaud, "From the bitstream to the netlist", Proceedings of ACM FPGA'08, 2008

[23] http://spectrum.ieee.org/semiconductors/processors/the-hidden-dangers-of-chopshop-electronics, 2013.