Chapter 1

SECURITY ASPECTS OF FPGAS IN CRYPTOGRAPHIC APPLICATIONS

Thomas Wollinger and Christof Paar *
Chair for Communication Security (COSY)
Ruhr-Universität Bochum, Germany
{wollinger, cpaar}@crypto.rub.de

Chapter in "New Algorithms, Architectures, and Applications for Reconfigurable Computing", editor Wolfgang Rosenstiel and Patrick Lysaght, Kluwer, 2004.

Abstract

This contribution provides a state-of-the-art description of security issues on FPGAs from a system perspectives. We consider the potential security problems of FPGAs and propose some countermeasure for the existing drawbacks of FPGAs. Even though there have been many contributions dealing with the algorithmic aspects of cryptographic schemes implemented on FPGAs, this contribution is one of the few investigations of system and security aspects.

Keywords: cryptography, security, attacks, reconfigurable hardware, FPGA, cryptographic applications, reverse engineering

1. Introduction and Motivation

In recent years, FPGAs manufacturers have come closer to filling the performance gap between FPGAs and ASICs, enabling them, not only to serve as fast prototyping tools but, also to become active players as components in systems. Reconfigurable hardware devices seem to combine the advantages of software and hardware implementations. Furthermore, there are potential advantages of reconfigurable hardware in cryptographic applications: algorithm agility, algorithm upload, architecture

*This research was partially sponsored by the German Federal Office for Information Security (BSI).

efficiency, resource efficiency, algorithm modification, throughput, cost efficiency.

The choice of the implementation platform of a digital system is driven by many criteria and heavily dependent on the application area. In addition to the aspects of algorithm and system speed and costs there are crypto-specific ones: physical security (e.g., against key recovery and algorithm manipulation); flexibility (regarding algorithm parameter, keys, and the algorithm itself); power consumption (absolute usage and prevention of power analysis attacks); and other side channel leakages.

The remainder of this chapter is organized as follows: We devoted the first part of this chapter to study FPGAs from a systems security perspective by describing some possible attacks (Section 2). In the second part we present possible countermeasures against the introduced attacks (Section 3). We end this contribution with some conclusions.

2. Shortcomings of FPGAs for cryptographic applications

This section summarizes security problems produced by attacks against given FPGA implementations. First we would like to state what the possible goals of such attacks are.

2.1 Why does someone wants to attack FPGAs?

The most common threat against an implementation of a cryptographic algorithm is to learn a confidential cryptographic key, that is, either a symmetric key or the private key of an asymmetric algorithm. Given that the algorithms applied are publicly known in most commercial applications, knowledge of the key enables the attacker to decrypt future (assuming the attack has not been detected and countermeasures have not been taken) and, often more harming, past communications which had been encrypted. Another threat is the one-to-one copy, or "cloning", of a cryptographic algorithm *together* with its key. In some cases it can be enough to run the cloned application in decryption mode to decipher past and future communications. In other cases, execution of a certain cryptographic operation with a presumingly secret key is in most applications the sole criteria which authenticates a communication party. An attacker who can perform the same function can masquerade as the attacked communication party. Yet another threat is given in applications where the cryptographic algorithms are proprietary. Even though such an approach is not wide-spread, it is standard practice in applications such as pay-TV and in government communications. In such scenarios it is already interesting for an attacker to reverse-engineer the

encryption algorithm itself. The associated key might later be recovered by other methods (e.g., bribery or classical cryptanalysis.) The discussion above assumes mostly that an attacker has physical access to the encryption device. Whether that is the case or not depends heavily on the application. However, we believe that in many scenarios such access can be assumed, either through outsiders or through dishonest insiders.

In the following we discuss vulnerabilities of modern FPGAs against such attacks. In areas were no attacks on FPGAs have been published, we tried to extrapolate from attacks on other hardware platforms, mainly memory cell and chip cards.

2.2 Description of the Black Box Attack

The classical method to reverse engineer a chip is the so called Black Box attack. The attacker inputs all possible combinations, while saving the corresponding outputs. The intruder is then able to extract the inner logic of the FPGA, with the help of the Karnaugh map or algorithms that simplify the resulting tables. This attack is only feasible if a small FPGA with explicit inputs and outputs is attacked and a lot of processor power is available. The reverse engineering effort grows and it will become less feasible as the size and complexity of the FPGA increases. The cost of the attack, furthermore, rises with the usage of state machines, LFSRs (Linear Feedback Shift Registers), integrated storage, and, if pins can be used, as input and output [8].

2.3 Cloning of SRAM FPGAs

The security implications that arise in a system that uses SRAM FPGAs are obvious, if the configuration data is stored unprotected in the system but external to the FPGA. In a standard scenario, the configuration data is stored externally in nonvolatile memory (e.g., PROM) and is transmitted to the FPGA at power up in order to configure the FPGA. An attacker could easily eavesdrop on the transmission and get the configuration file. This attack is therefore feasible for large organizations as well as for those with low budgets and modest sophistication.

2.4 Description of the Readback Attack

Readback is a feature that is provided for most FPGA families. This feature allows to read a configuration out of the FPGA for easy debugging. An overview of the attack is given in [8]. The idea of the attack is to read the configuration of the FPGA through the JTAG or programming interface in order to obtain secret information (e.g. keys, proprietary algorithm). The readback functionality can be prevented

with a security bit. In some FPGA families, more than one bit is used to disable different features, e.g., the JTAG boundary. In [4], the idea of using a security antifuse to prevent readout of information is patented.

However, it is conceivable, that an attacker can overcome these countermeasures in FPGA with fault injection. This kind of attack was first introduced in [7]. The authors showed how to break public-key algorithms, such as the RSA and Rabin signature schemes, by exploiting hardware faults. Furthermore, they give a high level description of transient faults, latent faults, and induced faults. This publication, was followed by [6], where the authors introduced differential fault analysis, which can potentially be applied against all symmetric algorithms in the open literature. Meanwhile there have been many publications that show different techniques to insert faults, e.g., electro magnetic radiation [22], infrared laser [2], or even a flash light [28]. It seems very likely that these attacks can be easily applied to FPGAs, since they are not especially targeted to ASICs. Therefore, one is able to deactivate security bits and/or the countermeasures, resulting in the ability to read out the configuration of the FPGA [16, 8].

Despite these attacks Actel Corporation [1] claims that after the programming phase, the cells of FPGAs cannot be read at all. On the other hand Xilinx offers the users the software tool JBits [9], which provides an API to access the bitstream information and allows dynamic reconfiguration for Xilinx Virtex FPGAs. JBits allows a simplified and automated access to specific part of the bitstream, resulting in a extra advantage for the attacker who performs a readback attack.

2.5 Reverse-Engineering of the Bitstreams

The attacks described so far output the bitstream of the FPGA design. In order to get the design of proprietary algorithms or the secret keys, one has to reverse-engineer the bitstream. The condition to launch the attack is not only that the attacker has to be in possession of the bitstream, but furthermore the bitstream has to be in the clear, meaning it is not encrypted.

FPGA manufactures claim, that the security of the bitstream relies on the disclosure of the layout of the configuration data. This information will only be made available if a non-disclosure agreement is signed, which is, from a cryptographic point of view, an extremely insecure situation. This security-by-obscurity approach was broken at least ten years ago when the CAD software company NEOCad reverse-engineered a Xilinx FPGA. NEOCad was able to reconstruct the necessary information about look-up tables, connections, and storage elements [26].

Hence, NEOCad was able to produce design software without signing non-disclosure agreements with the FPGA manufacturer. Even though a big effort has to be made to reverse engineer the bitstream, for large organizations it is quite feasible. In terms of government organizations as attackers, it is also possible that they will get the information of the design methodology directly from the vendors or companies that signed NDAs.

2.6 Description of Side Channel Attacks

Any physical implementation of a cryptographic system might provide a *side channel* that leaks unwanted information. Examples for side channels include in particular: power consumption, timing behavior, and electromagnet radiation. Obviously, FPGA implementations are also vulnerable to these attacks. In [17] two practical attacks, Simple Power Analysis (SPA) and Differential Power Analysis (DPA) were introduced. The power consumption of the device while performing a cryptographic operation was analyzed in order to find the secret keys from a tamper resistant device. The main idea of DPA is to detect regions in the power consumption of a device which are correlated with the secret key. Moreover, in some cases little or no information about the target implementation is required. Since their introduction, there has been a lot of work improving the original power attacks (see, e.g., relevant articles in [13]. There seems to be very little work at the time of writing addressing the feasibility of actual side channel attacks against FPGAs. Very recently the first experimental results of simple power analysis on an ECC implementation on an FPGA have been presented in [19] and on RSA and DES implementations in [30]. Somewhat related was the work presented in [27] which concludes that 60% of the power consumption in a XILINX Virtex-II FPGA is due to the interconnects and 14% and 16% is due to clocking and logic, respectively. These figures would seem to imply that and SPA type attack would be harder to implement on an FPGA than on an ASIC. However, the results presented in [30, 19] show that SPA attacks are feasible on FPGAs and that they can be realized in practice.

2.7 Description of Physical Attacks

The aim of a physical attack is to investigate the chip design in order to get information about proprietary algorithms or to determine the secret keys by probing points inside the chip. Hence, this attack targets parts of the FPGA, which are not available through the normal I/O pins. This can potentially be achieved through visual inspections and by using tools

such as optical microscopes and mechanical probes. However, FPGAs are becoming so complex that only with advanced methods, such as Focused Ion Beam (FIB) systems, one can launch such an attack. To our knowledge, there are no countermeasures to protect FPGAs against this form of physical threat. In the following, we will try to analyze the effort needed to physically attack FPGAs manufactured with different underlying technologies.

2.7.1 SRAM FPGAs. Unfortunately, there are no publications available that accomplished a physical attack against SRAM FPGAs. This kind of attack is only treated very superficially in a few articles, e.g. [23]. In the related area of SRAM memory, however there has been a lot of effort by academia and industry to exploit this kind of attack [10, 11, 3, 34, 25, 29, 18]. Due to the similarities in structure of the SRAM memory cell and the internal structure of the SRAM FPGA, it is most likely that the attacks can be employed in this setting.

Contrary to common wisdom, the SRAM memory cells do not entirely loose the contents when power is cut. The reason for these effects are rooted in the physical properties of semiconductors (see [11] for more details). The physical changes are caused mainly by three effects: electromigration, hot carriers, and ionic contamination.

Electromigration implies a high current density, that relocates metal atoms in the opposite direction of the current flow. Electromigration results in voids at the negative electrode and hillocks and whiskers at the positive electrode. The effect, that electrons with very high energy are able to overcome the $Si - SiO_2$ potential barrier and accelerate into the gate oxide are called hot carrier. This electrons stay there and it can take days until they neutralize [11]. Ionic contamination is triggered by the sodium ions present in the material used during semiconductor manufacturing and packaging process. Electrical fields and high temperature enable the move towards the silicon/sillicon-dioxide interface, resulting in a change of the threshold voltage.

In the published literature one can find several different techniques to determine the changes in device operations. Most publications agree that device can be altered, if 1) threshold voltage has changed by 100mV or 2) there is a 10% change in transconductance, voltage or current. An extreme case of data recovery, was described in [3]. The authors were able to extract a DES master key from a module used by a bank, without any special techniques or equipment on power-up. The reason being that the key was stored in same SRAM cells over a long period of time. Hence, the key was "burned" into the memory cells and the key values were retained even after switching off the device.

"I_{DDQ} testing" is one of the widely used methods and it is based on the analysis of the current usage of the device. The idea is to execute a set of test vectors until a given location is reached, at which point the device current is measured. Hot carrier effects, cell charge, and transitions between different states can then be detected at the abnormal I_{DDQ} characteristic [11, 34]. In [25], the authors use the substrate current, the gate current, and the current in the drain-substrate diode of a MOSFET to determine the level and duration of stress applied.

When it becomes necessary to access internal portions of a device, there are also alternative techniques available to do so, as described in [29]. Possibilities are to use the scan path that the IC manufacturers insert for test purposes or techniques like bond pad probing [11].

When it becomes necessary to use access points that are not provided by the manufacturer, the layers of the chip have to be removed. Mechanical probing with tungsten wire with a radius of $0,1 - 0,2\mu m$ is the traditional way to discover the needed information. These probes provide gigahertz bandwidth with $100fF$ capacitance and $1M\Omega$ resistance.

Due to the complex structure and the multi layer production of chips the mechanical testing is not sufficient enough. Focused Ion Beam (FIB) workstations can expose buried conductors and deposit new probe points. The functionality is similar to an electron microscope and one can inspect structures down to 5nm [18]. Electron-beam tester (EBT) is another measurement method. An EBT is a special electron microscope that is able to speed primary electrons up to 2.5 kV at 5nA. EBT measures the energy and amount of secondary electrons that are reflected.

Resulting from the above discussion of attacks against SRAM memory cells, it seems likely that a physical attack against SRAM FPGAs can be launched successfully, assuming that the described techniques can be transferred. However, the physical attacks are quite costly and having the structure and the size of state-of-the-art FPGA in mind, the attack will probably only be possible for large organizations, for example intelligence services.

2.7.2 Antifuse FPGAs. To discuss physical attacks against antifuse (AF) FPGAs, one has to first understand the programming process and the structure of the cells. The basic structure of an AF node is a thin insulating layer (smaller than $1\mu m^2$) between conductors that are programmed by applying a voltage. After applying the voltage, the insulator becomes a low-resistance conductor and there exists a connection (diameter about 100nm) between the conductors. The programming function is permanent and the low-impedance state will persist indefinitely.

In order to be able to detect the existence or non-existence of the connection one has to remove layer after layer, or/and use cross-sectioning. Unfortunately, no details have been published regarding this type of attack. In [8], the author states that a lot of trial-and-error is necessary to find the configuration of one cell and that it is likely that the rest of the chip will be destroyed, while analyzing one cell. The main problem with this analysis is that the isolation layer is much smaller than the whole AF cell. One study estimates that about 800,000 chips with the same configuration are necessary to explore the configuration file of an Actel A54SX16 chip with 24,000 system gates [8]. Another aggravation of the attack is that only about 2–5 % of all possible connections in an average design are actually used. In [23] a practical attack against AF FPGAs was performed and it was possible to alter one cell in two months at a cost of $1000. Based on these arguments some experts argue that physical attacks against AF FPGAs are harder to perform than against ASICs [1]. On the other hand, we know that AF FPGAs can be easily attacked if not connected to a power source. Hence, it is easier to drill holes to disconnect two connections or to repair destroyed layers. Also, depending on the source, the estimated cost of an attack and its complexity are lower [23].

2.7.3 Flash FPGAs. The connections in flash FPGAs are realized through flash transistors. That means the amount of electrons flowing through the gate changes after configuration and there are no optical differences as in the case of AF FPGAs. Thus, physical attacks performed via analysis of the FPGA cell material are not possible. However, flash FPGAs can be analyzed by placing the chip in a vacuum chamber and powering it up. The attacker can then use a secondary electron microscope to detect and display emissions. The attacker has to get access to the silicon die, by removing the packet, before he can start the attack [8]. However, experts are not certain about the complexity of such an attack and there is some controversy regarding its practicality [1, 23]. Other possible attacks against flash FPGAs can be found in the related area of flash memory. The number of write/erase cycles are limited to 10,000 – 100,000, because of the accumulation of electrons in the floating gate causing a gradual rise of the transistors threshold voltage. This fact increases the programming time and eventually disables the erasing of the cell [11]. Another less common failure is the programming disturbance in which unselected erased cells gain charge when adjacent selected cells are written [5]. This failure does not change the read operations but it can be detected with special techniques described in [11]. Furthermore, there are long term retention issues, like electron emission.

The electrons in the floating gate migrate to the interface with the underlying oxide from where they tunnel into the substrate. This emission causes a net charge loss. The opposite occurs with erased cells where electrons are injected [21]. Ionic contamination takes place as well but the influence on the physical behavior is so small that it can not be measured. In addition, hot carrier effects have a high influence, by building a tunnel between the bands. This causes a change in the threshold voltage of erased cells and it is especially significant for virgin cells [12]. Another phenomenon is overerasing, where an erase cycle is applied to an already-erased cell leaving the floating gate positively charged. Thus, turning the memory transistor into a depletion-mode transistor [11].

All the described effects change in a more or less extensive way the cell threshold voltage, gate voltage, or the characteristic of the cell. We remark that the stated phenomenons apply for EEPROM memory and that due to the structure of the FPGA cell these attacks can be simply adapted to attack flash/EEPROM FPGAs.

2.7.4 Summary of Physical Attacks. It is our position that due to the lack of published physical attacks against FPGAs, it is very hard (if at all possible) to predict the costs of such an attack. It is even more difficult to compare the effort needed for such an attack to a similar attack against an ASIC as there is no publicly available contribution which describes a physical attack against an FPGA that was completely carried out. Nevertheless, it is possible to draw some conclusions from our above discussion.

First, we notice that given the current size of state-of-the-art FPGAs, it seems unfeasible, except perhaps for large government organizations and intelligent services, to capture the whole bitstream of an FPGA. Having said that, we should caution that in some cases, an attacker might not need to recover the whole bitstream information but rather a tiny part of it, e.g., the secret-key. This is enough to break the system from a practical point of view and it might be feasible.

On the other hand, there are certain properties that *might* increase the effort required for a physical attack against FPGAs when compared to ASICs. In the case of SRAM-, Flash-, EPROM-, and EEPROM-FPGAs there is no printed circuit (as in the case of ASICs) and therefore it is potentially harder to find the configuration of the FPGA. The attacker has to look for characteristics that were changed on the physical level during the programming phase. In the case of antifuse FPGAs the effort for a physical attack *might* increase compared to ASICs because one has to find a tiny connection with a diameter of about 100 nm in a 1 μm^2

insulation layer. Furthermore, only $2 - 5\%$ of all possible connections are used in an average configuration.

3. Prevention of Attacks

This section shortly summarizes possible countermeasures that can be provided to minimize the effects of the attacks mentioned in the previous section. Most of them have to be realized by design changes through the FPGA manufacturers, but some could be applied during the programming phase of the FPGA.

3.1 How to prevent Black Box Attacks

The Black Box Attack is not a real threat nowadays, due to the complexity of the designs and the size of state-of-the-art FPGAs (see Section 2.2). Furthermore, the nature of cryptographic algorithms prevents the attack as well. Cryptographic algorithms can be segmented in two groups: symmetric-key and public-key algorithms. Symmetric-key algorithms can be further divided into stream and block ciphers. Today's stream ciphers output a bit stream, with a period length of 128 bits [32]. Block ciphers, like AES, are designed with a block length of 128 bits and a minimum key length of 128 bits. Minimum length in the case of public-key algorithms is 160 bits for ECC and 1024 bits for discrete logarithm and RSA-based systems. It is widely believed, that it is infeasible to perform a brute force attack and search a space with 2^{80} possibilities. Hence, implementations of this algorithms can not be attacked with the black box approach.

3.2 How to prevent cloning of SRAM FPGAs

There are many suggestions to prevent the cloning of SRAM FPGAs, mainly motivated by the desire to prevent reverse engineering of general, i.e., non-cryptographic, FPGA designs. One solution would be to check the serial number before executing the design and delete the circuit if it is not correct. This approach is not practical because of the following reasons: 1) The whole chip, including the serial number can be easily copied; 2) Every board would need a different configuration; 3) Logistic complexity to manage the serial numbers [16]. Another solution would be to use dongles to protect the design [14, 16]. Dongles are based on security-by-obscurity, and therefore do not provide solid security, as it can be seen from the software industry's experience using dongles for their tools. A more realistic solution would be to have the nonvolatile memory and the FPGA in one chip or to combine both parts by covering

them with epoxy. This reflects also the trend in chip manufacturing to have different components combined, e.g., the FPSLIC from Atmel. However, it has to be guaranteed that an attacker is not able to separate the parts.

Encryption of the configuration file is the most effective and practical countermeasure against the cloning of SRAM FPGAs. There are several patents that propose different scenarios related to the encryption of the configuration file: how to encrypt, how to load the file into the FPGA, how to provide key management, how to configure the encryption algorithms, and how to store the secret data In [36], the authors proposed that to partly decrypt the configuration file, in order to increase the debugging effort during the reverse engineering. If an attacker copies the partly decrypted file, the non-decrypted functionality is available, whereas the one decrypted is not. Thus, the attacker tries to find the errors in the design not aware of the fact, that they are caused through the encrypted part of the configuration. Most likely an attacker with little resources, would have dropped the reverse engineering effort, when realizing that parts are decrypted (which he did not do because he did not know). However, this approach adds hardly any extra complexity to an attack if we assume that an attacker has a lot of resources. In [15] an advanced scenario is introduced where the different parts of the configuration file are encrypted with different keys. The 60RS family from Actel was the first attempt to have a key stored in the FPGA in order to be able to encrypt the configuration file before transmitting it to the chip. The problem was that every FPGA had the same key on board. This implies that if an attacker has one key he can get the secret information from all FPGAs. In [14], the author discusses some scenarios where depending on the manufacturing cost, more than one key is stored in the FPGA.

An approach in a completely different direction would be to power the whole SRAM FPGA with a battery, which would make transmission of the configuration file after a power loss unnecessary. This solution does not appear practical, however, because of the power consumption of FPGAs. Hence, a combination of encryption and battery power provides a possible solution. Xilinx addresses this with an on-chip 3DES decryption engine in its Virtex II [35] (see also [20]), where only the two keys are stored in the battery powered memory. Due to the fact that the battery powers only a very small memory cells, the battery is limited only by its own life span.

3.3 How to prevent Readback Attacks

The readback attack can be prevented with the security bits set, as provided by the manufactures, see Section 2.4. If one wants to make sure that an attacker is not able to apply fault injection, the FPGA has to be embedded into a secure environment, where after detection of an interference the whole configuration is deleted or the FPGA is destroyed.

3.4 How to prevent Side Channel Attack

In recent years, there has been a lot of work done to prevent side-channel attacks, see [13]. The methods can generally be divide into software and hardware countermeasures, with the majority of proposals dealing with software countermeasures. "Software" countermeasures refer primarily to algorithmic changes, such as masking of secret keys with random values, which are also applicable to implementations in custom hardware or FPGA. Hardware countermeasures often deal either with some form of power trace smoothing or with transistor-level changes of the logic. Neither seem to be easily applicable to FPGAs without support from the manufacturers. However, some proposals such as duplicated architectures might work on today's FPGAs.

3.5 How to prevent Physical Attacks

To prevent physical attacks, one has to make sure that the retention effects of the cells are as small as possible, so that an attacker can not detect the status of the cells. Already after storing a value in a SRAM memory cell for 100–500 seconds, the access time and operation voltage will change [33]. Furthermore, the recovery process is heavily dependant on the temperature: 1.5 hours at $75°C$, 3 days at $50°C$, 2 month at $20°C$, and 3 years at $0°C$ [11]. The solution would be to invert the data stored periodically or to move the data around in memory. Cryptographic applications cause also long-term retention effects in SRAM memory cells by repeatedly feeding data through the same circuit. One example is specialized hardware that uses always the same circuits to feed the secret key to the arithmetic unit [11]. Neutralization of this effect can be achieved by applying an opposite current [31] or by inserting dummy cycles into the circuit [11]. In terms of FPGA application, it is very costly or even impractical to provide solutions like inverting the bits or changing the location for the whole configuration file. A possibility could be that this is done only for the crucial part of the design, like the secret keys. Counter techniques such as dummy cycles and opposite current approach can be carried forward to FPGA applications.

In terms of flash/EEPROM memory cell, one has to consider that the first write/erase cycles causes a larger shift in the cell threshold [24] and that this effect will become less noticeably after ten write/erase cycles [12]. Thus, one should program the FPGA about 100 times with random data, to avoid these effect (suggested for flash/EEPROM memory cells in [11]). The phenomenon of overerasing flash/EEPROM cells can be minimized by first programming all cells before deleting them.

4. Conclusions

This chapter analyzed possible attacks against the use of FPGA in security applications. Black box attacks do not seem to be feasible for state-of-the-art FPGAs. However, it seems very likely for an attacker to get the secret information stored in a FPGA, when combining readback and fault injection attacks. Cloning of SRAM FPGA and reverse engineering depend on the specifics of the system under attack, and they will probably involve a lot of effort, but this does not seem entirely impossible. Physical attacks against FPGAs are very complex due to the physical properties of the semiconductors in the case of flash/SRAM/EEPROM FPGAs and the small size of AF cells. It appears that such attacks are even harder than analogous attacks against ASICs. Even though FPGA have different internal structures than ASICs with the same functionality, we believe that side-channel attacks against FPGAs, in particular power-analysis attacks, will be feasible too.

It seems from our previous remarks that, while, the art of cryptographic algorithm implementation is reaching maturity, FPGAs as a security platform are not, and in fact, that they might be currently out of question for security applications. We don't think that is the right conclusion, however. It should be noted that many commercial ASICs with cryptographic functionality are also vulnerable to attacks similar to the ones discussed here. A commonly taken approach to prevent these attacks is to put the ASIC in a secure environment. A secure environment could, for instance, be a box with tamper sensors which triggers what is called "zeroization" of cryptographic keys, when an attack is being detected. Similar approaches are certainly possible for FPGAs too. (Another solution often taken by industry is not to care and to build cryptographic products with poor physical security, but we are not inclined to recommend this.)

References

[1] Actel Corporation. Design Security in Nonvolatile Flash and Antifuse. Avaialble at http://www.actel.com/appnotes/DesignSecurity.pdf, August 2002.

[2] C. Ajluni. Two New Imaging Techniques to Improve IC Defect Indentification. *Electronic Design*, 43(14):37–38, July 1995.

[3] R. Anderson and M. Kuhn. Low Cost Attacks on Tamper Resistant Devices. In *5th International Workshop on Security Protocols*, pages 125–136. Springer-Verlag, 1997. LNCS 1361.

[4] J. M. Aplan, D. D. Eaton, and A. K. Chan. Security Antifuse that Prevents Readout of some but not other Information from a Programmed Field Programmable Gate Array. United States Patent, Patent No. 5898776, April 27 1999.

[5] S. Aritome, R. Shirota, G. Hemink, T. Endoh, and F. Masuoka. Reliability Issues of Flash Memory Cells. *Proceedings of the IEEE*, 81(5):776–788, May 1993.

[6] E. Biham and A. Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In *Advances in Cryptology - CRYPTO '97*, pages 513–525. Springer-Verlag, 1997. LNCS 1294.

[7] D. Boneh, R. A. DeMillo, and R. J. Lipton. On the Importance of Checking Cryptographic Protocols for Faults (Extended Abstract). In *Advances in Cryptology - EUROCRYPT '97*, pages 37–51. Springer-Verlag, 1997. LNCS 1233.

[8] B. Dipert. Cunning circuits confound crooks. http://www.e-insite.net/ednmag/contents/images/21df2.pdf, October 12 2000.

[9] S. A. Guccione and D. Levi. Jbits: A java-based interface to fpga hardware. Technical report, Xilinx Corporation, San Jose, CA, USA, 2003. Available at http://www.io.com/ guccione/Papers/Papers.html.

[10] P. Gutmann. Secure Deletion of Data from Magnetic and Solid-State Memory. In *Sixth USENIX Security Symposium*, pages 77–90, July 22-25, 1996.

[11] P. Gutmann. Data Remanence in Semiconductor Devices. In *10th USENIX Security Symposium*, pages 39–54, August 13–17, 2001.

[12] S. Haddad, C. Chang, B. Swaminathan, and J. Lien. Degradations due to hole trapping in flash memory cells. *IEEE Electron Device Letters*, 10(3):117–119, March 1989.

[13] B. S. Kaliski, Jr., Ç. K. Koç, D. Naccache, C. Paar, and C. D. Walter, editors. *CHES 1999-2003*, Berlin, Germany, September 2003. Springer-Verlag. LNCS 1717/1965/2162/2523/2779.

[14] T. Kean. Secure Configuration of Field Programmable Gate Arrays. In *International Conference on Field-Programmable Logic and Applications 2001 (FPL 2001)*, pages 142–151. Springer-Verlag, 2001. LNCS 2147.

[15] S. H. Kelem and J. L. Burnham. System and Method for PLD Bitstram Encryption. United States Patent, Patent Number 6118868, September 12 2000.

[16] D. Kessner. Copy Protection for SRAM based FPGA Designs, May 8 2000. Available at http://www.free-ip.com/copyprotection.html.

[17] P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *CRYPTO '99*, pages 388–397. Springer-Verlag, 1999. LNCS 1666.

[18] O. Kommerling and M. Kuhn. Design Principles for Tamper-Resistant Smartcard Processors. In *USENIX Workshop on Smartcard Technology (Smartcard '99)*, pages 9–20, May 1999.

[19] S. Örs, E. Oswald, and B. Preneel. Power-Analysis Attacks on an FPGA — First Experimental Results. In *CHES 2003*, pages 35–50. Springer-Verlag, 2003. LNCS 2779.

[20] R. C. Pang, J. Wong, S. O. Frake, J. W. Sowards, V. M. Kondapalli, F. E. Goetting, S. M. Trimberger, and K. K. Rao. Nonvolatile/ battery-backed key in PLD. United States Patent, Patent Number 6366117, Nov. 28 2000.

[21] C. Papadas, G. Ghibaudo, G. Pananakakis, C. Riva, P. Ghezzi, C. Gounelle, and P. Mortini. Retention characteristics of single-poly EEPROM cells. In *European Symposium on Reliability of Electron Devices, Failure Physics and Analysis*, page 517, October 1991.

[22] J.-J. Quisquater and D. Samyde. Electro Magnetic Analysis (EMA): Measures and Countermeasures for Smart Cards. In *International Conference on Research in Smart Cards, E-smart 2001*, pages 200 – 210, Cannes, France, September 2001.

[23] G. Richard. Digital Signature Technology Aids IP Protection. In EETimes - News, 1998. Available at http://www.eetimes.com/news/98/1000news/digital.html.

[24] K. San, C. Kaya, and T. Ma. Effects of erase source bias on Flash EPROM device reliability. *IEEE Transactions on Electron Devices*, 42(1):150–159, January 1995.

[25] D. Schroder. *Semiconducor Material and Device Characterization.* John Wiley and Sons, 2nd edition, 1998.

[26] G. Seamann. FPGA Bitstreams and Open Designs. Available at http://www.opencollector.org/news/Bitstream, 2000.

[27] L. Shang, A. Kaviani, and K. Bathala. Dynamic Power Consumption on the Virtex-II FPGA Family. In *2002 ACM/SIGDA 10th International Symposium on Field Programmable Gate Arrays*, pages 157–164. ACM Press, 2002.

[28] S. Skorobogatov and R. Anderson. Optical Fault Induction Attacks. In *CHES 2002*, pages 2–12. Springer-Verlag, 2002. LNCS 2523.

[29] J. Soden and R. Anderson. IC failure analysis: techniques and tools for quality and reliability improvement. *Proceedings of the IEEE*, 81(5):703–715, May 1993.

[30] F.-X. Standaert, L. van Oldeneel tot Oldenzeel, D. Samyde, and J.-J. Quisquater. Power Analysis of FPGAs: How Practical is the Attack. In *13th International Conference on Field Programmable Logic and Applications — FPL 2003*. Springer-Verlag, 2003. LNCS 2778.

[31] J. Tao, N. Cheung, and C. Ho. Metal Electromigration Damage Healing Under Bidirectional Current Stress. *IEEE Transactions on Elecron Devices*, 14(12):554–556, December 1993.

[32] S. Thomas, D. Anthony, T. Berson, and G. Gong. The W7 Stream Cipher Algorithm. Available at http://www.watersprings.org/ pub/id/draft-thomas-w7cipher-03.txt, April 2003. Internet Draft.

[33] J. van der Pol and J. Koomen. Relation between the hot carrier lifetime of transistors and CMOS SRAM products. In *International Reliability Physics Symposium (IRPS 1990)*, page 178, April 1990.

[34] T. Williams, R. Kapur, M. Mercer, R. Dennard, and W. Maly. IDDQ Testing for High Performance CMOS - The Next Ten Years. In *IEEE European Design and Test Conference (ED&TC'96)*, pages 578–583, 1996.

[35] Xilinx Inc. Using Bitstream Encryption. Handbook of the Virtex II Platform, 2003. Available at http://www.xilinx.com.

[36] K.-W. Yip and T.-S. Ng. Partial-Encryption Technique for Intellectual Property Protection of FPGA-based Products. *IEEE Transactions on Consumer Electronics*, 46(1):183–190, 2000.