

Dynamically Configurable Security for SRAM FPGA Bitstreams

Lilian Bossuet, Guy Gogniat, Wayne Burleson

▶ **To cite this version:**

Lilian Bossuet, Guy Gogniat, Wayne Burleson. Dynamically Configurable Security for SRAM FPGA Bitstreams. International Journal of Embedded Systems, 2006, 2, pp.73 - 85. hal-00089394

HAL Id: hal-00089394
https://hal.science/hal-00089394

Submitted on 18 Oct 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Dynamically configurable security for SRAM FPGA bitstreams

Lilian Bossuet* and Guy Gogniat

LESTER laboratory of Université de Bretagne Sud,
56321 Lorient, France
Fax: 332 97 87 45 27 E-mail: bossuet@ixl.fr
E-mail: guy.gogniat@univ-ubs.fr
*Corresponding author

Wayne Burleson

Electrical and Computer Engineering Department,
University of Massachusetts, Amherst, MA 01003, USA
Fax: 413-545-1993 E-mail: burleson@ecs.umass.edu

Abstract: FPGAs are becoming increasingly attractive – thanks to the improvement of their capacities and their performances. Today, FPGAs represent an efficient design solution for numerous systems. Moreover, since FPGAs are important for electronic industry, it becomes necessary to improve their security, particularly for SRAM FPGAs, since they are more vulnerable than other FPGA technologies. This paper proposes a solution to improve the security of SRAM FPGAs through flexible bitstream encryption. This proposition is distinct from other works because it uses the latest capabilities of SRAM FPGAs like partial dynamic reconfiguration and self-reconfiguration. It does not need an external battery to store the secret key. It opens a new way of application partitioning oriented by the security policy.

Keywords: field programmable gate arrays; design security; bitstream encryption; partial reconfiguration and self-reconfiguration; reconfigurable architecture.

Reference to this paper should be made as follows: Bossuet, L., Gogniat, G. and Burleson, W. (xxxx) 'Dynamically configurable security for SRAM FPGA bitstreams', *Int. J. Embedded Systems*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Lilian Bossuet is a PhD student in Electrical Engineering in LESTER laboratory of the Université de Bretagne Sud, Lorient France, where he has been since 2001. He has a BSEE from the ENSEA Cergy-Pontoise, France. He has a MSEE from the INSA-Université de Rennes, France. He has succeeded an Electrical Engineering competitive examination for teacher training from ENS Cachan. He has worked as a Tool Engineer for Autoliv Electronics. He was a Visitor (summer 2003) at the University of Massachusetts Amherst USA. His researches are in reconfigurable computing and particularly design space exploration for coarse-grained reconfigurable architecture. He also conducts research in high methodology and tools for SoC, FPGA utilisation and performances estimation, FPGA security.

Guy Gogniat is an Associate Professor of Electrical and Computer Science at the University of Bretagne Sud, Lorient, France, where he has been since 1998. He has a BSEE from the FIUPSO Orsay, France and a MSEE from the University of Paris Sud Orsay and a PhD in ECE from the University of Nice-Sophia Antipolis France. His researches are in the general area of CAD and reconfigurable computing, including codesign methodologies and software radio platform exploration with funding from national research projects (AS, RNTL, RNRT) and national and international companies and organisations (CNRS, CEA, THALES, MITSUBISHI …). He also conducts research in high-level methodologies and tools for FPGA utilisation, performance estimation and FPGA security.

Wayne Burleson is an Associate Professor of Electrical and Computer Engineering at the University of Massachusetts Amherst where he has been since 1990. He has a BSEE and MSEE from MIT and a PhD in ECE from the University of Colorado. His research is in the general area of VLSI and Signal Processing, including circuits for low-power, long interconnects, clocking and mixed signals with funding from NSF, SRC, Compaq/HP and Intel. He also conducts research in reconfigurable computing, content-adaptive signal processing, smart cards and multimedia instructional technologies. He is a member of the ACM, ASEE, Sigma Xi, a senior member of the IEEE Society.

Copyright © 200x Inderscience Enterprises Ltd.

1 Introduction

The FPGA (Field Programmable Gate Array) concept was born during the 80s, when the configuration point size (transistors or fuses) was too large in comparison with the chip size to have an interesting FPGA density. Therefore, these devices were just used to do prototyping or glue logic. For a long time, the FPGAs have not taken benefit from the best deep-submicronic technology, today the more advanced FPGAs use 90 nanometer technology with copper metallisation (best actual accessible technology). With the improvement of technological processes and since the FPGAs structure is very regular, it is possible to build some FPGAs with more than one million transistors. Thanks to these evolutions, FPGAs are increasingly attractive for numerous systems and to build efficient SoC (System on a Chip). The FPGAs market continues to increase and FPGAs are capturing the classical market share of ASIC (Application Specific Integrated Circuit) market. The cost crossover point, which permits to know the necessary number of systems built to choose an efficient ASIC solution, is increasingly far (Tredennick and Shimamoto, 2003). It is possible even for a large number of systems built to choose an economically efficient FPGA solution.

Since FPGAs are becoming so important for the electronic industry, it is necessary to think about the security of FPGA-based systems. It is possible to consider the FPGA-based systems' security problem in three ways.

1.1 Security system using FPGA

In this case, FPGA is used as a part of the security system. The FPGA dynamic reconfiguration improves the security system's flexibility. Therefore, it is possible to change the classical software update by hardware update in order to prevent attacks evolutions.

For example, internet-connected hosts are now frequently attacked by malicious machines located around the world. Hosts can be protected from remote machines by filtering the traffic through a firewall. Use of an FPGA can be very efficient for such application in order to build less static system. In Lockwood et al. (2003) a System-On-Programmable-Chip (SOPC), internet firewall has been implemented that protects high-speed networks from present and future threats. The high level of flexibility and extensibility required by such systems is guaranteed by the use of an FPGA (in Lockwood et al. (2003) authors use a Xilinx Virtex FPGA).

In the same way, in Dandalis and Prasanna (2000), the authors use Xilinx FPGA to develop an Adaptive Cryptographic Engine (ACE) for Internet Protocol Security (IPSec) architectures. Several FPGA configurations of cryptographic algorithms are stored in a memory in the form of cryptographic library. The FPGA is configured on-demand based on the cryptographic library and then performs the required encryption/decryption tasks.

We think that it is also possible to use the FPGA concept (e.g., reconfiguration, hardware update) for smart cards system or PAY-TV, for example. However, today, there is no published work on these applications.

1.2 Protecting FPGA data

In this case, it is necessary to protect the application that runs on FPGA. The data inside the circuit and the data transferred to/from the peripheral circuits during the communication must be protected. The main solution is to integrate data encryption scheme inside the FPGA. These circuits are attractive for executing the actual cryptographic algorithms and are of particular importance from security point of view. There has been a large amount of work done dealing with the algorithmic and computer architecture aspects of cryptographic schemes implemented on FPGA over the last five years. According to Wollinger et al. (2004) and Wollinger and Paar (2003), we can list the potential advantages of FPGA in cryptographic applications.

- *Algorithm agility.* This term refers to the cryptographic algorithms switching during operation of the targeted application. While algorithm agility is costly with traditional hardware, FPGA can be reprogrammed on the fly.
- *Algorithm upload.* It is perceivable that fielded devices are upgraded with a new encryption algorithm. FPGA-equipped encryption devices can upload the new configuration code.
- *Architecture efficiency.* In certain cases hardware architecture can be much more efficient if it is designed for a specific set of parameters. An example for the parameters for cryptographic algorithms can be the key. FPGA allows this type of devices and optimisations with a specific parameter set. Owing to the nature of FPGA, the application can be changed totally or partially.
- *Resource efficiency.* The majority of security protocols are hybrid protocols that need several algorithms. As they are not used simultaneously, the same FPGA device can be used for both through run-time reconfiguration.
- *Algorithm modification.* There are applications that require modification of standardised cryptographic algorithms.
- *Throughput.* General-purpose microprocessors are not optimised for fast execution. Although, typically slower than ASIC implementations, FPGA implementations have the potential of running substantially faster than software implementations (as with a processor).

- *Cost efficiency.* There are two cost factors, which have to be taken into consideration when analysing the cost efficiency of FPGAs: cost of development and unit price. The costs to develop an FPGA implementation of a given algorithm are much lower than that for an ASIC implementation. The unit prices are not significant when compared with the developmental costs. However, for high-volume applications (more than one million of circuit build) ASIC solution usually becomes the more cost-efficient choice.

1.3 FPGA design security

In this last case, the protection concerns the design against cloning and reverse engineering. It is custom intellectual property protection. Concerning the SRAM FPGAs, the design security corresponds to the way to protect the bitstream or the FPGA configuration.

This paper focuses on the latter case dealing with FPGA design security. If the FPGA design itself is not secure, the other security problems cannot be efficiently treated. Using an unsecured device embedded in a security system is not security-efficient. Many works already proposed solutions to protect the bitstream. However, the contribution of this paper relies on the utilisation of the latest improvements of SRAM FPGAs configuration techniques to answer the security problem.

This paper is organised as follows. Section 2 describes some aspects of the design security problem such as the classical hardware devices security level. Section 3 presents several works dealing with the protection of SRAM FPGA configuration. Section 4 describes the new capability of SRAM FPGA self-reconfiguration. In Section 5 a new SRAM FPGA bitstream protection solution is proposed. The drawbacks and advantages of the proposed solutions are given in Section 6. Section 7 compares the different solutions of design security for FPGA. Finally, Section 5 concludes this paper and exposes several future directions.

2 Design security

It is interesting, before investigating the different solutions to secure the configuration of SRAM FPGAs, to list what are the different attacks against an integrated circuit today, what is the protection level of some current circuits and why do they have this level of protection?

2.1 Need for design security

The problem of design security is simple; the designer does not want a competitor to be able to pirate his design. There are two sorts of piracy.

- *Cloning.* When a competitor makes an exact copy of a design including the board layout and chip, and when he is able to create a copy of the pirated system.
- *Reverse engineering*. When a competitor copies a design by reconstructing a 'schematic' or net list level representation. In this process, he analyses and understands how the design works and how to improve it, or to modify it with malicious intents. Reverse engineering generally consists of the following stages:
  - analysis of the product
  - generation of an intermediate level product description
  - human analysis of the product description to produce a specification
  - generation of a new product using the specification.

Therefore, reverse engineering is more serious than cloning. These two aspects correspond to different attacks, and the design security must protect the system against both attacks. To perform cloning or reverse engineering, two types of attack can be considered; the non-invasive and the invasive attacks.

The non-invasive attacks gather all the methods that use external means. For example, the attackers can use all the possibilities of the circuit inputs in order to obtain all the different outputs and draw the system truth table; this method is called 'Black Box Attack'.

In the case of SRAM FPGA, a simple attack method is intercepting the bitstream between the root ROM and the FPGA when the system power is switched on. More complex attacks can be brought into play; time, power and electromagnetic changes and measures like the simple or differential power analysis – interested readers can refer to the works on power analysis of FPGA in Standaert et al. (2003, 2004) and Örs et al. (2003).

The invasive attacks (or *physical attacks*) are characterised by the necessity to destroy the integrated circuit (component package) to study the chip (design inside the component) with some complex methods. For example, it is possible to use laser cutter microscope in order to split the chip in several slices and understand the chip layout. These attacks can use sophisticated tools like optical microscope, mechanical probes and even Focused Ion Beam (FIB). As these attacks use the weakness of the silicon technology, when they are possible, it is very hard to secure the system against them.

The paper Anderson and Kuhn (1996, 1997) give some information about these different attacks. It is possible to classify the integrated circuits according to their protection against the different types of attacks. The next section presents an example of security level classification.

2.2 Protection level of some circuits

The level of protection offered by actual integrated circuits is an interesting metric to identify works that must be carried out to improve the security level of one particular type of integrated circuit. In the IBM Systems Journal, a paper Abraham et al. (1991) defines the various security

levels for modern electronic systems and the corresponding taxonomy of attackers.

- *Level 0 (ZERO)*. No special security features added to the system. It is easy to comprise the system with low cost tools.
- *Level 1 (LOW)*. Some security features in place. They are relatively easily defeated with common laboratory or shop tools.
- *Level 2 (MODLOW)*. The system has some security against non-invasive attacks; it is protected for some invasive attacks. More expensive tools are required, as well as specialised knowledge.
- *Level 3 (MOD).* The system has some security against non-invasive and invasive attacks. Special tools and equipment are required, as well as some special skills and knowledge. The attack may become time-consuming but will eventually be successful.
- *Level 4 (MODH).* The system has strong security against attacks. Equipment is available but is expensive to buy and operate. Special skills and knowledge are required to use the equipment for an attack. More than one operation may be required so that several adversaries with complementary skills would have to work on the attack sequence. The attack could be unsuccessful.
- *Level 5 (HIGH).* The security features are very strong. All known attacks have been unsuccessful. Some research by a team of specialists is necessary. Highly specialised equipment is necessary, some of which might have to be designed and built. The success of the attack is uncertain.

According to this classification, it is possible to give a general security level for the current integrated circuits. Of course, these different levels are not fixed and depend of the factory and the type of circuit (in the same factory there are several families and some of them can be especially security-efficient like some military families). The authors have tried to give one level by classical integrated circuit and explain the reason of their choices. The security level of the classical integrated circuits is given in Table 1.

Table 1 Security level of classical integrated circuits

Integrated circuit	*Security level*
Conventional SRAM FPGA	0
ASIC gate array	3
Cell-based ASIC	3
SRAM FPGA with bitstream encryption	3
Flash FPGA	4
Antifuse FPGA	4

Conventional SRAM FPGAs have the lowest security level. These circuits need a bitstream transfer from the root ROM at power up (because the memory of configuration is a SRAM volatile memory). Therefore, it is easy for the pirate to read with a simple probe the bitstream during the transfer. The conventional SRAM FPGAs are inefficient for safe design. However, with a bitstream encryption it is possible to clearly improve the security level since the security weakness is secure. SRAM FPGAs have a good resistance against some attacks like power analysis (Standaert et al., 2003). Today few works present the results of attacks against SRAM FPGA (Örs et al., 2003 and Standaert et al., 2004).

Often considered like a secure technology, ASICs are actually relatively easy to reverse engineer. Because, unlike FPGAs, ASICs do not have switch. Therefore, it is possible to strip the chip to copy with certitude the complete layout in order to understand how it works. Methods to reverse engineer ASIC exist. The cost of reverse engineering is high since the tools required are expensive and the process is time consuming. Therefore, it is not a simple process and therefore the security level is 3 for such devices.

Contrary to the ASICs, the FPGAs, like antifuse or flash, are actually security-efficient since they are based on switches. With these FPGAs, no bitstream can be intercepted in the field (no bitstream transfer, no external configuration device). In the case of antifuse FPGAs, the attacker needs a Scanning Electron Microscope (SEM) in order to know the state of each antifuse. Nevertheless, the difference between a programming and a non-programming antifuse is very difficult to see. Moreover, such analysis is intractable in a device like Actel AX2000 that contains 53 million of antifuses and according to Actel (www.actel.com/products/rescenter/security/index.html) only 2–5% (average) of these antifuses are programmed. For flash FPGA, there is no optical difference after configuration, so the invasive attacks are very complex. The same advantages are given by QuickLogic to promote their flash FPGAs with the ViaLink technology (QuickLogic, 2002).

If the antifuse and the flash FPGAs are very security-efficient, they are just one time configurable (or one time programmable), so they are not reconfigurable devices. The system build with these devices, is not flexible. If the designer wants a reconfigurable device, he must target a SRAM FPGA. Moreover, the capacities of the SRAM FPGAs are the highest for FPGA devices. Actually, the SRAM FPGAs have a market share higher than 60% (just with the two leaders companies Xilinx (http://www.xilinx.com) and Altera (http://www.altera.com)). Therefore, the research to improve the security level of such FPGAs and particularly the improvement of bitstream encryption is necessary today.

Some works give efficient solutions to encrypt the SRAM FPGA bitstream. Nevertheless, there are some drawbacks and it is possible to improve them taking into account the latest innovations of these FPGAs. The following section presents some works about the bitstream encryption.

3 Related work

Two approaches are generally possible to address the design security problem. The first one considers that the

best solutions to protect the devices against piracy are legal solutions. The definition of efficient laws, the regulation and the management of intellectual properties are parts of this solution.

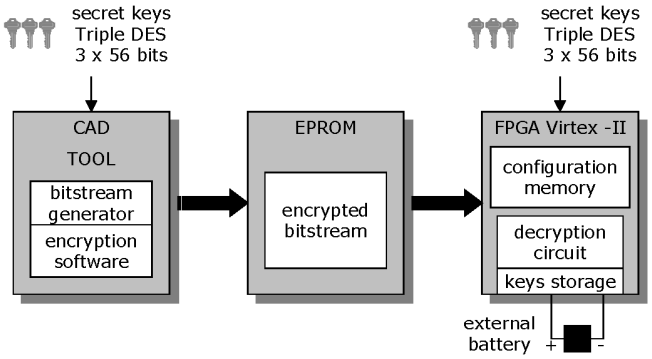
The second one, according to the last section, proposes to improve the security level of actual SRAM FPGAs by configuration protection (bitstream encryption). Even if the two solutions must be complementary, in the following, we only address the latter approach.

Xilinx proposes a security system (www.xilinx.com) based on a triple DES encryption scheme to protect the bitstream of the Virtex-II and Virtex-II Pro family device.

Xilinx CAD software tool encrypts the bitstream using the powerful Triple Data Encryption (DES) algorithm before downloading the configuration inside the FPGA. Triple DES is the standard used by many governments for safe communication and by banks around the world for money transfers. This algorithm uses three 56-bits public keys. The designer can use random keys or choose their own-keys.

Figure 1 shows the encryption/decryption system used by Xilinx to protect the configuration of Virtex-II devices.

Figure 1 Xilinx Virtex-II triple DES encryption scheme. The bitstream is encrypted by the CAD tool during the EPROM storage. When power is switched on, a DES decryption circuit embedded in the FPGA decrypts the configuration. Three 56-bits keys are embedded in the FPGA and stored in a volatile memory with an external battery



This system is relatively simple; it is just necessary to choose one option during the last step of the CAD process, the bitstream generation. First, a key file that describes the configuration of the three keys is programmed inside the FPGA. The customer chooses his own keys. Of course, it is not necessary to store the key file inside the configuration memory. It is not possible to encrypt two cores with different keys loaded into the same FPGA at the same time. The keys are stored in a dedicated SRAM memory inside the FPGA that can be backed up with a small battery (like a watch battery).

Next, the configuration step is performed like a classical configuration without the bitstream encryption. In fact, the configuration stored in the external EPROM is encrypted. The FPGA contains a decryption circuit that automatically detects when the bitstream is encrypted and it decrypts the configuration before the SRAM bits are programmed. Xilinx does not give information about the necessary extra-time to decrypt the configuration.

The Xilinx bitstream encryption scheme is efficient because without the correct key it is not possible to configure other chips with the encrypted bitstream. Nevertheless, when the device is configured, it is not possible to use partial reconfiguration or to do read-back and it is not possible to use bitstream compression.

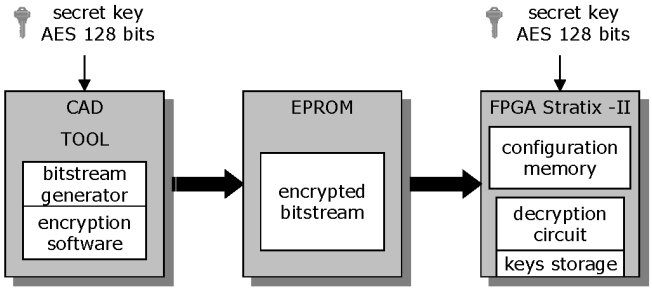
If the designer does not need security, the device can be configured with non-encrypted bitstream and the on-chip keys are simply ignored.

This method has a strong drawback; it uses an external battery to save the key. It is poor for several reasons. This solution costs a lot of area on the board and even if the used battery is small it is necessary to add a socket, and the board area is a critical issue for embedded system. Moreover, this solution increases the board cost (2–3$ per board (Trimberger, 2004)) and reduces the system lifetime (particularly bad for long-life hardware applied in space applications, for example).

It is necessary to improve the Xilinx solution by proposing a solution without the additional battery.

Not long ago, Altera proposed a solution of bitstream encryption for the new Stratix-II device (Altera Coporation, 2004). Figure 2 shows the encryption/decryption system used by Altera to protect the configuration of Stratix-II devices.

Figure 2 Altera Stratix-II AES encryption scheme. Like Xilinx solution, the bitstream is encrypted by the CAD tool during the EPROM storage. When power is switched on, an AES decryption circuit embedded in the FPGA decrypts the configuration. One 128-bits key is embedded in the FPGA and stored in a non-volatile memory without an external battery



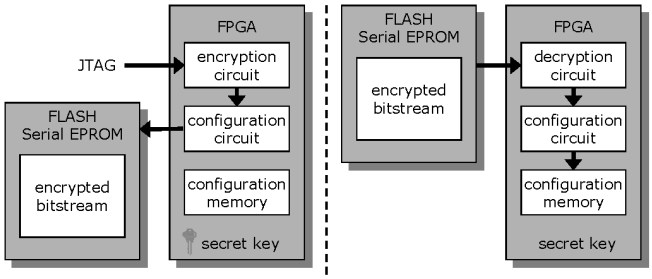
Design security in Stratix-II device is enabled by encrypting the configuration bitstream using 128-bit AES and a non-volatile key. AES is a standard for encryption, developed to replace the DES standard. The 128-bits AES key makes it much more secure than DES (56-bits key size) and triple DES (three 56-bits key). Unlike Xilinx solution, the non-volatile key retains its information when the power is off, eliminating the need for a backup battery.

Tom Kean of the Algotronix society proposes an attractive solution to answer the FPGA security problem (Kean, 2001; Kean et al., 2001). The first idea of Kean is to use a secret cryptographic key stored on an FPGA like Altera solution. He gives some ways to store this key as using a laser to program a set of links during manufacture.

As the secret key is only known by the FPGA, it must contain an encryption and a decryption circuit. However, contrary to Xilinx and Altera methods, the CAD does not change and just generates a classical bitstream.

Figure 3 shows the encryption/decryption system used by Kean to protect the configuration of SRAM FPGA. Figure 3(a) shows the initial configuration of secure FPGA and Figure 3(b) shows the normal configuration of secure FPGA.

Figure 3 Kean proposes encryption/decryption scheme embedded in the FPGA. (a) shows the initial configuration to encrypt the bitstream (inside the FPGA) and stores it in the EPROM and (b) shows the normal configuration of the FPGA when the power is switched on, the encrypted bitstream is decrypted inside the FPGA and configures it



This solution has many advantages; it does not affect system reliability, requires no additional components and it does not require support from CAD software. In this system, nobody (the designer or the CAD tool) needs knowledge of the key.

If Kean's and Altera solutions overcome the battery limitation of the Xilinx solution, all the solutions have the same important disadvantages. In all the cases, the decryption circuit is embedded inside the FPGA. These circuits take FPGA silicon area normally reserved for the developed application. Therefore, the total application dedicated-area is reduced by these solutions, particularly in the case of Algotronix solution, since the encryption and the decryption circuits are both embedded in the same FPGA.

Moreover, in all solutions the encryption and the decryption circuits are fixed, so it is not possible to upgrade them or to choose the encryption/decryption algorithm and architecture. It is a lack of flexibility for the system; it will be impossible to update it with new encryption algorithms, for example.

In all solutions, the entire design is encrypted with the same encryption algorithm. However, such approach is very restrictive since it does not consider any security policy. Actual designs (owing to the high degree of application complexity) are based on numerous heterogeneous parts that do not present the same 'security sensitivity'. Hence, the designer may want to partition his application in several parts and use different encryption/decryption algorithms to encrypt/decrypt these parts. For example, if the designer uses some free or very-easy-to-find IPs (Intellectual Property), it may be not necessary to encrypt these parts of the application. Other parts like interfaces, for example, do not need a high security level. On the other hand, the real designer's IPs need a high security level.

Finally, the three proposed solutions give only one fixed answer to the bitstream security problem and lack flexibility.

Other solutions are proposed; most of them can be found in recent US Patents for example, Kelen and Burnham. (2000), Erickson et al. (2001), Mason et al. (2001) and Pang et al. (2002). Nevertheless, these solutions are not very different from Xilinx (www.xilinx.com), Altera (www.altera.com) or Kean (2001; Kean et al., 2001) solutions.

If existing solutions are not very different one from another, it is mainly owing to the fact that they do not use the new features of SRAM FPGAs like partial reconfiguration, dynamic reconfiguration and self-reconfiguration.

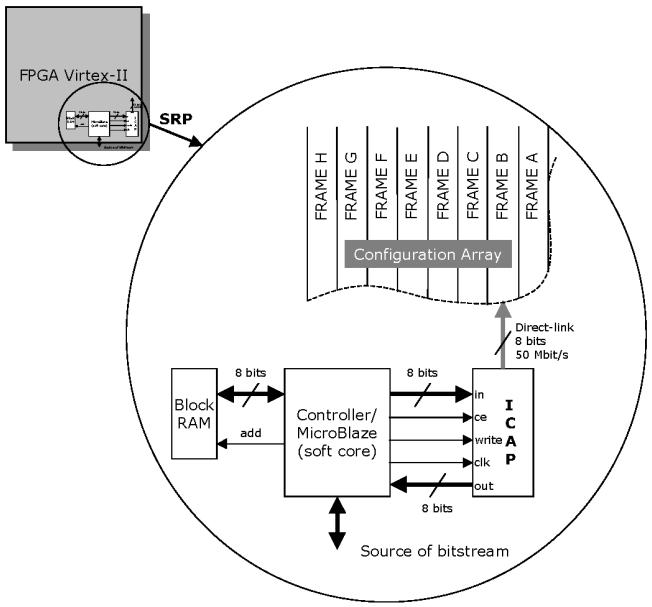
In the following section, we present the new self-reconfiguration capabilities of SRAM FPGA.

4 New self-reconfiguration technique for SRAM FPGA

According to previous sections, actual solutions to secure the SRAM FPGA bitstream are efficient, but lack flexibility. However flexibility, given by the reconfiguration capabilities, is the main advantage of the reconfigurable devices like SRAM FPGAs (particularly in comparison with other FPGAs or ASIC). This advantage is increasingly important with the new capabilities of SRAM FPGAs like partial reconfiguration, dynamical reconfiguration or self-reconfiguration.

In Blodget et al. (2003) and Blodget and McMillan (2003) Xilinx presents a Self-Reconfiguring Platform (SRP) for Xilinx Virtex-II and Xilinx Virtex-II Pro devices. Self-reconfiguration extends the concept of dynamic reconfiguration. It assumes that dedicated circuits within the FPGA are used to control the configuration of the other parts of the FPGA. In this case, the FPGA is able to dynamically reconfigure itself under the control of an embedded microprocessor or controller. This microprocessor can be a soft-core like Xilinx Micro Blaze (32-bit RISC) or a hard-core like IBM PowerPC (32-bit RISC) embedded on the Xilinx Virtex-II Pro. To perform the dynamical reconfiguration, the microprocessor or the controller use a specific interface called ICAD (Internal Configuration Access Port). When the bitstream is stored within the FPGA, the FPGA embedded RAM (called BlockRAM in Xilinx Virtex devices) are used like small configuration cache. Figure 4 presents a schematic view of the self-reconfigurable platform.

Figure 4 Schematic view of the self-reconfigurable platform SRP. The ICAP port is directly connected to the configuration array. It can partial reconfigure the different frame of configuration. The configuration controller can be a MicroBlaze soft core. The bitstream file can be provided from outside or inside the circuit. The BlockRAM can be used like configuration memory



The Virtex ICAP is a version of the Xilinx Select Map programming port that is internally accessible to the configure FPGA logic. According to Fong et al. (2003) the ICAP, interface is fairly simple, consisting of separate eight-bit datapaths for reads and writes, write and chip enables, a busy signal and a clock input. The ICAP interface is physically located in the lower right corner of the Virtex-II FPGA, and can be seen using the Xilinx FPGA editor tool. When using the Select Map express configuration mode (data available every clock cycle), ICAP can be loaded with data without the need for handshaking. The ICAP throughput is limited to 50 Mbit/s.

Xilinx proposes a tool to manage these new FPGA capabilities called XPART for Xilinx Partial Reconfiguration Toolkit.

Some applications of self-reconfiguration have been done in Fong et al. (2003), Ulmann et al. (2004) and Hübner et al. (2004). In Ulmann et al. (2004), self-reconfiguration is used for CAN-bus management, and in Hübner et al. (2004) the same authors use self-reconfiguration and bitstream compression.

In the following section, we present how the new solution to address the bitstream security problem takes advantage of the dynamic SRAM FPGA self-reconfiguration.

5 A new solution to protect the SRAM FPGA bitstream

5.1 Introduction

This solution takes benefit of the new possibilities of reconfiguration of SRAM FPGAs to improve their security level without the drawbacks highlighted previously.

The encryption and the decryption circuit must leave all the silicon area free for the developed application.

The solution must use an embedded key in order to work without an extra battery; to store the key, a model close to Kean's solution (Abraham et al., 1991; www.actel.com/products/rescenter/security/index.html) can be chosen. It is possible to use laser to engrave the key or use some antifuse elements to do a non-volatile key programming.

A very important feature is also to give the designer the opportunity to choose the encryption/decryption algorithms and architectures. In this way, it is possible to adapt the encryption/decryption scheme according to the requested security level for the developed application. Furthermore, this feature enables to easily upgrade the system if a new efficient encryption/decryption algorithm is available.

Finally, we address the security-sensitivity policy problem by allowing the designer to use different encryption algorithms for a single application. The Security-Critical Parts (SCP) of the application will only be encrypted.

For test, we use a Xilinx Virtex-II Pro XCV2VP20 FF1152 proto-board.

5.2 Application security policy

As the encryption/decryption scheme is costly owing to time, power consumption and takes silicon area, it is very interesting to adapt it according to the required security level of the application parts.

All the solutions presented in Section 3 use a complete bitstream encryption with a single encryption algorithm. Nevertheless, a security application analysis can show that some parts of the application do not need protection whereas other parts need strong protection. These last parts can be security-sensitive part (global system security) or they can be the custom intellectual properties with high development cost, for example. We call these application parts Security-Critical Parts (SCP) and the other parts, like some communication protocol IPs or easy-to-find IPs, the No-Critical Parts (NCP).

The designer must partition his application in function of the security level of the different parts. It is a security-oriented partitioning. He must choose the suitable encryption/decryption algorithm and architecture for the protection of the SCP bitstreams. The designer can choose

different encryption/decryption algorithms and architectures for several SCPs or he can choose the same for all. We think that it could be more security-efficient to choose different security features for the different SCPs.

To understand our approach, in the following, two examples are given; process during the initial configuration step and process during the normal configuration step of the FPGA. In the examples, the application is partitioned into three different parts; two SCPs that need high security level (they are encrypted with two different encryption algorithms) and one NCP that does not need encryption. For the examples, each SCP bitstream is encrypted with a different algorithm but a solution with a same algorithm can be considered. The case with two SCPs is just an example and other configurations can be considered.

5.3 Key management

One feature is very important in our solution; the key management. It is mandatory that a pirate cannot access the keys used by the different decryption/encryption circuits. To prevent spy configuration, we use bitstream authentication with checksum. The circuit used to control the bitstream authentication is embedded in the FPGA on the JTAG port.

Moreover, since in this solution, the decryption/ encryption algorithm is not fixed, it is necessary to store a large key. Indeed different algorithms do not use the same key size (for example the AES algorithm uses a 128-bits key, and the triple DES uses three 56-bits keys). In fact, among the *n*-key bits, the encryption/decryption circuits select *m* necessary bits. Since only the designer knows the position in the large key of the *m* chosen bits, it is a supplementary security barrier. With the large key knowledge necessary, the pirate must investigate to identify the effective key bits for the suitable algorithm.

5.4 Security configuration controller

Our solution uses the partial configuration and the dynamic self-reconfiguration of the FPGA. The management of such configuration process is complex, particularly for the self-reconfiguration. Moreover, several bitstreams are used while the application runs. In our system, there are three types of bitstream,

- encrypted bitstream of a SCP
- no-encrypted bitstream of a decryption circuit
- no-encrypted bitstream of a NCP.

The controller must be able to detect the different bitstreams. A bitstream signature (ID) gives the controller the bitstream characteristics (encrypted or not for example). These characteristics are used like processor instructions by the controller. According to the characteristics, the controller partial-configures directly the FPGA with the selected NCP bitstream or it partial-configures the FPGA with first the decryption circuit bitstream associated with an encrypted SCP bitstream before using self-reconfiguration to configure the FPGA with the decrypted SCP bitstream.

The security configuration controller is based on a finite state machine to perform the configuration management. To handle the configuration sequence, the controller needs the external EPROM memory partitioning (the memory mapping). We can notice that this mapping can be complex in order improve the system security. For example the designer can interleave the data stored in the memory and mix the several encrypted and no-encrypted configurations. A configuration address register stores the memory mapping.

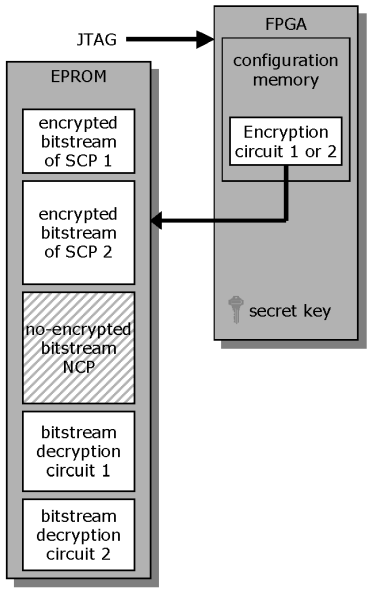
The security configuration controller can be external like a dedicated CPLD or a microprocessor. However, it is also possible that this controller is embedded inside the FPGA, like in the case of Xilinx self-configuration system (Fong et al., 2003). In this last case, the configuration controller can be a soft-core microprocessor (like Xilinx MicroBlaze) or a hard-core microprocessor (like IBM PowerPC for Xilinx VirtexII-Pro devices).

5.5 Initial FPGA configuration

The initial FPGA configuration is performed in the laboratory or manufactory in order to store all the different bitstreams in the EPROM memory. The CAD tool performs the initial configuration. If there are SCPs in the application, the bitstreams of each encryption circuits are generated to use these circuits to encrypt the SCPs bitstreams. In the same way, the bitstreams of the decryption circuits are generated to use these circuits to decrypt the encrypted SCPs bitstreams.

Figure 5 presents the encryption system when the FPGA is initially configured and the root configuration memory is programmed (initial configuration).

Figure 5 Encryption scheme during the initial FPGA configuration. The bitstreams are stored in the EPROM from the CAD tool through the FPGA JTAG port. For the SCPs bitstreams, the FPGA is configured with encryption circuits to encrypt the bitstreams before being stored it in the EPROM. The NCP bitstream are not encrypted.



During the initial FPGA configuration, the first step consists of programming the root configuration memory with the non-encrypted parts. First, the NCP bitstreams are stored; in the example shown in Figure 5, there is only one NCP. For the same example, two decryption circuits will be used to decrypt the encrypted SCPs bitstreams. Therefore, the bitstreams of the two decryption circuits are stored in the EPROM. In Figure 5, after the first step there are three no-encrypted bitstreams stored in the EPROM; the *NCP* bitstream, the *decryption circuit* 1 bitstream (associated with the *SCP* 1) and the *decryption circuit* 2 bitstream (associated with the *SCP* 2).

The second step is the storage of the encrypted bitstreams of the *SCP* 1 and *SCP* 2. First, it is necessary to configure the FPGA with the *encryption circuit* 1 in order to encrypt the bitstream of the *SCP* 1. Once the *SCP* 1 bitstream is encrypted, it is stored in the root external EPROM. Since the *SCP* 2 needs other encryption circuit, it is not necessary to keep the *encryption circuit* 1 in the device. The FPGA is partial configured with the *encryption circuit* 2; the *SCP* 2 bitstream is encrypted and stored in the EPROM.

Of course, it is necessary for the CAD to manage partial reconfiguration like in Xilinx proposition (Blodget and McMillan, 2003).

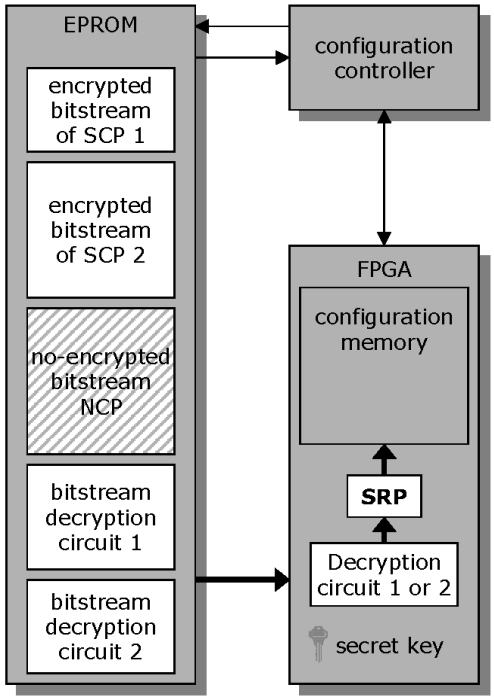
At the end of the initial configuration step, the root configuration memory contains the encrypted bitstreams of *SCP* 1 and *SCP* 2, the no-encrypted bitstream of the *NCP* and the no-encrypted bitstreams of the decryption circuits required to decrypt *SCP* 1 and *SCP* 2 (*decryption circuit* 1 and *decryption circuit* 2).

5.6 Normal FPGA configuration (when power is switched on)

When power is switched on, the SRAM FPGA must be configured since this inside configuration memory is volatile. Figure 6 shows the decryption-configuration system when the FPGA is configured from an external EPROM memory that stores the configuration (normal configuration). The configuration controller manages the configuration process.

The FPGA configuration process works as follows: First, the FPGA is configured with the *decryption circuit* 1 bitstream. Then the FPGA uses it to decrypt the encrypted *SCP* 1 bitstream and self-configures the *SCP* 1. As we can see on Figure 6, the SRP (Self-Reconfiguring Platform, see Section 4) is used to perform self-reconfiguration. Once the *SCP* 1 bitstream is decrypted and the FPGA is configured with the *SCP* 1 circuit, it is not necessary to keep the *decryption circuit* 1. The *decryption circuit* 2 replaces (with FPGA partial reconfiguration) it in order to decrypt the encrypted bitstream of *SCP* 2. In the same way, after the decryption and the self-configuration of the *SCP* 2 bitstream, it is not necessary to keep the *decryption circuit* 2.

Figure 6 Decryption and self-configuration scheme during the normal FPGA configuration. The FPGA is partial-configured by the decryption circuit 1 or 2 to decrypt the encrypted SCP 1 and SCP 2 bitstreams. The FPGA is self-configured with these decrypted bitstreams. The self-reconfiguration is performed by the SRP. At the end of the configuration process, the FPGA is configured with the NCP bitstream



After this first phase, the FPGA is configured with the *SCP* 1 and the *SCP* 2 circuits. The last step consists in configuring the FPGA free area with the other application parts that have not an encrypted configuration; so with the *NCP* bitstream.

Finally, the FPGA is configured with all the application parts; the *SCP* 1, the *SCP* 2 and the *NCP*. There can be any encryption or decryption circuit configured in the FPGA.

5.7 Configuration controller finite state machine

As described previously, the configuration controller is developed with a finite state machine. With the knowledge of the memory mapping, the configuration management finite state machine is relatively simple. The configuration controller is used only for normal FPGA configuration when power is switched on. The initial configuration is processed by the CAD tool.

Figure 7 shows the three-global-states used by the configuration controller. Table 2 describes the actions associated to the states of the configuration controller. The first state of this three-states FSM is an idle state. To change state the configuration controller waits for a start signal. This signal is the begin-signal of the normal configuration process.

Figure 7 Configuration controller finite state machine. It is a three-global-states machine. The states represent several actions. The active state depends if the bitstream is encrypted or not

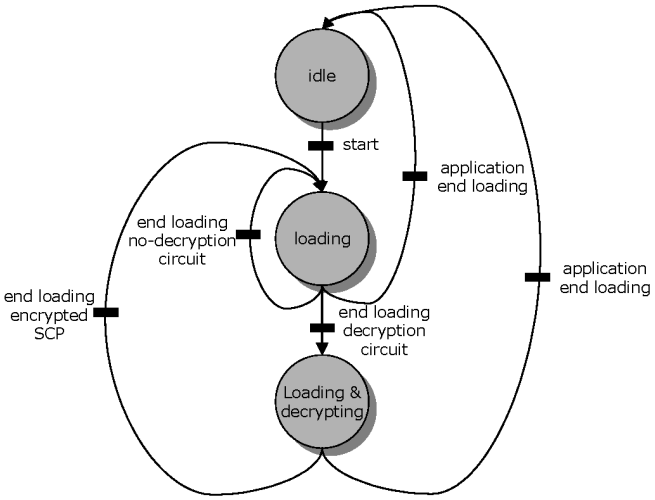


Table 2 States description of the configuration controller FSM

State name	*Actions*
Idle	Wait start
Loading	Configure the FPGA with selected bitstream* using partial-configuration
	Update the configuration address register
	*The selected bitstream can be the no-encrypted bitstream of a decryption circuit or a NCP
Loading and decrypting	Start the decryption algorithm and load the corresponding SCP bistream* on the FPGA
	Update the configuration address register
	*The selected bitstream is an encrypted bitstream

Once in the second state, the loading state, the configuration controller changes states according to the type of bitstream. If the bitstream is not encrypted the current state is the second state. In this state, the normal configuration of the FPGA is performed. If the bitstream is encrypted (so it is a SCP bitstream), the current state is the 'loading and decrypting' state. In this state, the configuration controller loads first the decryption circuit bitstream inside the FPGA before loading the encrypted bitstream of a SCP.

The machine returns to the idle state when all the application is loaded inside the FPGA.

This section has shown the main technological characteristics of our bitstream protection system for SRAM FPGA. The following sections give the drawbacks and advantages of our solution and compare it with the different solutions (presented in Section 3).

6 Drawbacks and advantages of the proposed solution

If this method permits to overcome the limitation of other proposed solutions, it has, however, some drawbacks.

The first drawback is the relative complexity of the method, since it is necessary to manage the partial reconfiguration and dynamic self-configuration. Most of the FPGA manufacturers do not have the technology and the CAD tools to manage these types of configurations but Xilinx, which proposes an efficient tool for such needs.

The decryption circuit can have several sizes according to the algorithm and the implementation. For example, several works give comparisons of the hardware performance of the different AES final candidates (MARS, RC6, Rijndael, Serpent or Twofish for example) using FPGA (Dandalis et al., 2000; Elbirt et al., 2000; Gaj and Chodowiec, 2000; Weaver and Wawrzynek, 2000). All these works use the Xilinx Virtex as the reconfigurable target. The Tables 3 and 4 compare the results of these studies for the area requirement (one Virtex slice corresponds to two four-inputs LUTs, two flip-flops and one carry chain) and time performance (throughput).

Table 3 Area requirement of FPGA implementations of AES final candidates

	No. of slices of the cryptographic core		
Algorithm	*Dandalis et al. (2000)*	*Elbirt et al. (2000)*	*Gaj and Chodowiec (2000)*
Rijndael	4312	5302	2902
Serpent	1250	7964	4438
RC6	1749	3189	1139
Twofish	2809	3053	1076
MARS	4621	–	2737

Table 4 Time performance of FPGA implementations of AES final candidates

	Throughput (Mbit/s)		
Algorithm	*Dandalis et al. (2000)*	*Elbirt et al. (2000)*	*Gaj and Chodowiec (2000)*
Rinjdael	353.0	300.1	331.5
Serpent	148.9	444.2	339.4
RC6	112.9	126.5	103.9
Twofish	173.1	119.6	177.3
MARS	101.9	–	39.8

The performances (time and area) showed in the two tables are different for each work. Because the architectures chosen, for the different studies, have different structures (loop unroll, pipeline and sub-pipeline). All these results are given only for an encryption core without the key-setup circuit. Nevertheless, this circuit must be considered because it can take area (slices). The Table 5 shows the number of slices for key-setup circuit of the five AES final candidates and the relative area percentage of the total area requirement (encryption core and key-setup circuits).

According to these results, it is significant to consider the key-setup circuit in the area requirement. Finally, the three tables show that a same decryption standard (AES in this example) can be performed with several

algorithms and each algorithm can have different implementations. Therefore, it is necessary to give all the possibilities to the designer, and our solution gives this flexibility. Moreover, the studies Dandalis et al. (2000), Elbirt et al. (2000), Gaj and Chodowiec (2000) and Weaver and Wawrzynek (2000) are throughput oriented, therefore, the area (or the number of used FPGA resources) is not the main constraint. In our system, the out data of the decryption circuit are used to self-reconfigure the FPGA. In the case of Xilinx technology, the ICAP interface limits the throughput to 50 Mbit/s. This throughput is widely inferior to most of the Table 4 results. Therefore, it is possible to develop decryption algorithm with area (used resources) constraint. Actually, the number of Virtex slices used for the cryptographic cores given in Table 3 must be reduced. For example, in our Xilinx proto-board, the Virtex-II Pro XC2VP20 contains 9280 slices, according to the Table 3, with such device, the Rijndael implementation of AES use from 31% to 57%. It is probably necessary to limit the number of used resources since the FPGA is not configured only with the decryption algorithm.

Table 5 Area requirement of FPGA implementations of AES final candidates

	No. of slices of the key-setup circuit		*Percent of the total area*	
Algorithm	*Dandalis et al. (2000)*	*Weaver and Wawrzynek (2000)*	*Dandalis et al. (2000)*	*Weaver and Wawrzynek (2000)*
Rijndael	1361	128	24	14
Serpent	1300	2060	51	35
RC6	901	290	34	15
Twofish	6554	1260	70	48
MARS	2275	50	33	3

The configuration controller can be complex. Its complexity depends on the number of SCPs in the application. This number is correlated to the application security partitioning. The costs of a larger root memory and a complex configuration controller are the hardware overhead costs of this method but they represent the origin of its flexibility. The system security has always costs that are necessary to evaluate in order to choose the best solution according to the required security level.

Since it is necessary to first configure the decryption circuit before the real configuration of each SCP, this method can spend time when the system is powered up. Nevertheless, today the SRAM FPGA configuration is increasingly faster (about 10 millisecond for a partial reconfiguration for a Xilinx Virtex 1000-E device (Delahaye et al., 2004)).

This method has many very interesting advantages. First, the encryption/decryption circuits do not take FPGA application-dedicated resources, since when a decryption circuit has been used it is removed from the FPGA. The FPGA resources initially used to perform the decryption circuit are free for other uses.

We choose, like Kean (2001; Kean et al 2001), to embed the key inside the FPGA in order to have non-external extra-battery.

One of the main advantages of this method is the increase of flexibility. The designer can partition the application according to the required security level. Therefore, if just a small part of the application needs a strong security, the system can be very simple (just one small SCP). The designer has the possibilities to choose the suitable algorithms and architectures for the encryption/ decryption circuits. It is possible to adjust the security level according to the application constraints.

Moreover, the designer can upgrade his application and the security scheme with the same reconfigurable hardware. In this way, it is possible to take advantage of the latest improvements of the security field.

7 Comparison of the different actual solutions

Section 3 of this paper has shown different actual solutions of FPGA protection against cloning and reverse engineering. It is interesting to compare these different solutions with our solution for several aspects; security level, encryption flexibility, reconfiguration flexibility and complexity. Table 6 presents the result of comparisons.

Table 6 Area requirement of FPGA implementations of AES final candidates

	Security	*Encryption flexibility*	*Reconfiguration flexibility*	*Complexity*
Actel Antifuse	High	–	Any	Easy
QuickLogic Flash	High	–	Any	Easy
Xilinx Triple DES	Middle	Any	Low	Easy
Altera AES	Middle	Any	Low	Easy
Algotronix T. Kean	Middle	Any	Low	Middle
UBS/UMASS L. Bossuet	Middle+	High	High	Complex

According to the table, we think that the security level is higher for antifuse or flash logic, but we think that it is necessary to better expertise the real security level of bitstream encryption system. The real advantage of our solution is the flexibility of encryption and reconfiguration. Moreover, with a real application security policy (i.e., security-oriented application partitioning), our solution proposes a higher security level that the other solution for SRAM FPGA. Nevertheless, our solution complexity is higher since it is necessary to manage the partial and self-reconfiguration.

8 Conclusion

Since the SRAM FPGAs are increasingly important for the electronic industry, it is necessary to improve the security level of such devices. Although some works have already proposed solutions to improve this security level, we think that is it possible to investigate more this domain.

In this paper, we propose a new solution to prevent piracy against SRAM FPGAs bitstream. Our contribution is to use the latest developments of configuration technique in order to improve the security system flexibility. The use of self-reconfiguration allows using the decryption circuit out data to configure the decrypted bitstream. Unlike the actual bitstream encryption scheme (Xilinx or Altera solution), our solution is flexible; the designer can choose the different encryption/decryption algorithms and architectures. He can easily update the system with new security feature. Moreover, we propose to the designer to apply a true security policy for the applications, by security-oriented partitioning.

We think that the security problem is a very important issue for FPGAs and for the reconfigurable systems on chip. Probably in the near future, there will be more and more works done about this subject.

Acknowledgement

Manuscript received June 28, 2004. This work was supported in part by the French Ministry for Education and Research.

References

Abraham, D.G., Dolan, G.M., Double, G.P. and Stevens, J.V. (1991) 'Transaction security system', *IBM Systems Journal*, Vol. 30, No. 2, pp.206–229.

Altera Coporation (2004) *Design Security in Stratix II Devices*, White paper, Available on www.altera.com.

Anderson, R. and Kuhn, M. (1996) 'Tamper resistance – a cautionary note', *Proceeding of the Second USENIX Workshop on Electronic Commerce*, November 18–21, Oakland, California, USA, pp.1–11.

Anderson, R. and Kuhn, M. (1997) 'Low cost attack on tamper resistant devices', *Proceeding of the 5th Workshop of Security Protocols*, April 7–9, Paris, France, pp.125–136.

Blodget, B. and McMillan, S. (2003) 'A lightweight approach for embedded reconfiguration of FPGAs', *Design, Automation and Test in Europe Conference and Exhibition*, DATE'03, Mach 3–7, Munich, Germany.

Blodget, B., James-Roxby, P., Keller, E., McMillan, S. and Sundararajan, P. (2003) 'A self-reconfiguration platform', *Proceeding of 13th International Conference on Field-Programmable Logic and Applications*, FPL'2003, September, Lisbon, Portugal, pp.565–574.

Dandalis, A. and Prasanna, V.K. (2000) 'An adaptive cryptographic for IPSec architectures', *Proceeding IEEE Symposium on Field-Programmable Custom Computing Machines*, FCCM'00, April, Napa, USA, pp.132–141.

Dandalis, A., Prasanna, K. and Rolim, J.D.P. (2000) 'A compartive study of performances of the AES final candidates using FPGA', *Workshop on Cryptographic Hardware and Embedded Systems*, August.

Delahaye, J.P., Gogniat, G., Roland, C. and Bomel, P. (2004) 'Software radio and dynamic reconfiguration on a DSp/FPGA platform', in Rykaczewski, P. and Schmidt, M. (Eds.): in special issue on *Software Defined Radio of Frequenz*, May-June, No. 58, pp.152–159.

Elbirt, A.J., Yip, W., Chetwynd, B. and Paar, C. (2000) 'An FPGA implementation and performance evaluation of the AES block cipher candidate algorithm finalists', *Proceeding of the third Advanced Encryption Standard Candidate Conference*, AES3, April 12–14, New York, USA, pp.12–27.

Erickson, C.R., Tovana, D. and Holen, V.A. (2001) *Encryption of Configuration Stream*, US Patent 6 212 639, April 3.

Fong, R., Harper, S. and Athanas, P. (2003) 'A versatile framework for FPGA field updates: an application of partial self-reconfiguration', *Proceeding of 14th IEEE International Workshop on Rapid System Prototyping*, RSP'03, 9–11 June, San Diego, California, USA, pp.117–123.

Gaj, K. and Chodowiec, P. (2000) 'Comparison of the hardware performance of the AES candidates using reconfigurable hardware', *Proceeding of the third Advanced Encryption Standard Candidate Conference*, AES3, April 12–14, New York, USA.

Hübner, M., Ulmann, M., Weissel, F. and Becker, J. (2004) 'Real-time configuration code decompression for dynamic FPGA self-reconfiguration', *11th IEEE Reconfigurable Architectures Workshop*, RAW 2004, Santa Fé, New Mexico, USA, 26–17 April.

Kean, T. (2001) 'Secure configuration of field programmable gate array', *Proceedings IEEE Symposium on Field Programmable Custom Computing Machines (FCCM)*, Rohnert Park CA.

Kean, T. (2001) 'Secure configuration of field programmable gate arrays', *Proceeding of 11th International Conference on Field-Programmable Logic and Applications*, FPL'2001. Belfast, UK, pp.142–152.

Kelen, S.H. and Burnham, J.L. (2000) *System and Method for PLD Bitstream Encryption*, US Patent 6 118 869, September 12.

Lockwood, J.W., Neely, C., Zuver, C., Moscola, J., Dharmapurikar, S. and Lim, D. (2003) 'An extensible, system-on-programmable-chip, content-aware internet firewall', *Proceeding of 13th International Conference on Field-Programmable Logic and Applications*, FPL'2003, September, Lisbon, Portugal, pp.859–868

Mason, M.T., Kunnari, N.D. and Kuo, H.H. (2001) *Secure Programmable Logic Device*, US Patent 6 331 784, December 18.

Örs, S.B., Oswald, E. and Preneel, B. (2003) 'Power-analysis attack on an FPGA – first experimental results', *CHES 2003, LNCS 2779*, pp.35–50.

Pang, R.C., Wong, J., Frake, S.O., Sowards, J.W., Kondapalli, V. M., Goetting, F.E., Trimberger, S.M. and Rao, K.K. (2002) *Non Volatile/Battery-Backed Key in PLD*, US Patent 6 336 117, April 2.

QuickLogic (2002) *Security in QuickLogic Devices*, White Paper, Available on http://www.quicklogic.com.

Standaert, F.X., Örs, S.B. and Preneel, B. (2004) 'Power-analysis on an FPGA implementation of AES', In Joye, M. and Quisquarter, J.J. (Eds.): *Proceedings of Cryptographic Hardware and Embedded Systems CHES'2004, Lecture Note in Computer Science (LNCS)*, Springer-Verlag, pp.30–44.

Standaert, F.X., van Oldeneel tot Oldenzeel, L., Samyde, D. and Quisquater, J.J. (2003) 'Power analysis of FPGAs: how practical is the attack', *Proceeding of 13th International Conference on Field-Programmable Logic and Applications*, FPL'2003, September, Lisbon, Portugal, pp.707–711.

Tredennick, N. and Shimamoto, B. (2003) 'The rise of reconfigurable systems', *Proceeding of Engineering of Reconfigurable Systems and Application*, ERSA'2003, June 23–26, Las Vegas, Nevada, USA, pp.3–9.

Trimberger, S. (2004) 'Virtex encrypted bitstreams', *2nd International Workshop on Cryptographic Architectures Embedded in Reconfigurable Devices*, CryptArchi 2004, Dijon, France, June 16–18.

Ulmann, M., Hübner, M., Grimm, B. and Becker, J. (2004) 'An FPGA run-time system for dynamical on-demand reconfiguration', *11th IEEE Reconfigurable Architectures Workshop*, RAW 2004, Santa Fé, New Mexico, USA, 26–17 April.

Weaver, N. and Wawrzynek, J. (2000) 'A comparison of the AES candidates amenability to FPGA implementation', *Proceeding of the third Advanced Encryption Standard Candidate Conference*, AES3, April 12–14, New York, USA.

Wollinger, T. and Paar, C. (2003) 'How secure are FPGAs in cryptographic applications', *Proceeding of 13th International Conference on Field-Programmable Logic and Applications*, FPL'2003, September, Lisbon, Portugal, pp.707–711.

Wollinger, T., Guajardo, J. and Paar, C. (2004) 'Security on FPGAs, state of the art implementations and attacks', *ACM Transactions in Embedded Computing Systems (TECS)*, Vol. 3, No. 3, pp.534–574.

Websites

Actel Coporation, Resource Center: Security, Available on www.actel.com/products/rescenter/security/index.html.

Altera Coporation, http://www.altera.com.

Xilinx Coporation, http://www.xilinx.com.

Xilinx Coporation, *Virtex-II platform FPGA Handbook*, Technical Documentation, Available on www.xilinx.com.