Securing Embedded Programmable Gate Arrays in Secure Circuits

Nicolas Valette[1,2], Lionel Torres[2], Gilles Sassatelli[2] and Frédéric Bancel[1]

[1]STMicroelectronics
Smartcard Division
ZI de Rousset, 13 106 Rousset Cedex, France
{nicolas.valette, frederic.bancel}@st.com

[2]LIRMM / UMR 5506
Microelectronics Department
161 rue Ada, 34 492 Montpellier Cedex 5, France
{valette, torres, sassatelli}@lirmm.fr

Abstract

The purpose of this article is to propose a survey of possible approaches for implementing embedded reconfigurable gate arrays into secure circuits. A standard secure interfacing architecture is proposed and motivations justifying such an approach are discussed. This paper also lists all features offered by FPGA vendors (Field Programmable Gate Array) aiming at securing those circuits according to different concerns. This article emphasizes on configuration memory programming which is probably the weakest point of using programmable devices on a secure context.

1. Introduction

Secure circuits are considered as the main elements for performing the security of a given system. For instance, they are used to store secret keys, to control confidential data access or to encrypt/decrypt secure data [1]. For performing these operations, secure circuits are mainly composed of a CPU (Central Process Unit), non-volatile and volatile memories and cryptographic operators. Representative secure circuit architecture is described in Figure 1.

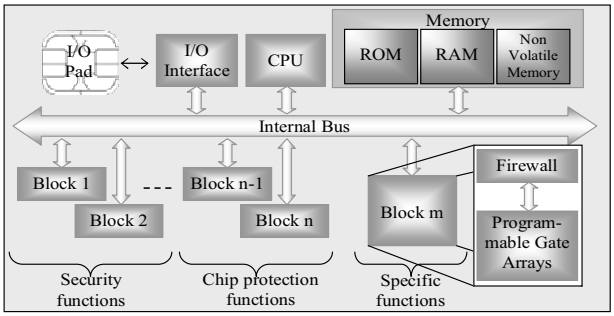

Figure 1. Secure component architecture.

Application fields for such systems are rapidly widening and are no longer limited to banking transactions. Health, electronic passport, pay TV or mobile phones are a few examples where providing security (under different forms: confidentiality, authenticity, integrity) is a must. Some application fields need a high bandwidth communication interface, others require important storage memory. Matching both system specifications and security concerns often implies the use of hardware solutions.

Manufacturers also have to integrate in their chip specific countermeasures against potential attacks. With the improvement of technologies and communications media, especially the Internet, hackers become stronger in bypassing security functions. [2] provides an interesting overview on what hackers can do. Authors present different ways to obtain secret data hidden in circuits: invasive or non invasive attacks. Some types of non invasive attacks are based on side channels [3]. Those techniques rely on the analysis of information that leaks from the circuit like power consumption, electromagnetic emanations, temperature or even sound ... SPA (Simple Power Analysis) and DPA (Differential Power Analysis) are common attacks carried out on smartcards. Countermeasures and chip protection functions are embedded into secure chips to defeat those attacks.

From a financial point of view, secure devices tend to become cheaper, chip vendors have to reduce their manufacturing costs. Therefore, in order to lower the production cost as much as possible, they generally design dedicated applications for a customer including minimal features. Another solution consists in creating a standard product with a complete feature set corresponding to every single customer's requirements. This solution of course tends to be too expensive. Additionally, dedicated products often lack minimum flexibility, which prevent customers from updating their products.

A considered solution for addressing these problems in secure circuits relies on using embedded reconfigurable logic. This approach is expected to offer more flexibility and performance compared to CPU based devices. It should also enable the improvement of security through being more reactive and resistive to hacker's threats. For example, a circuit update founded on a different cryptographic algorithm could void a given attack.

Integration of a reconfigurable core into chips has already been done with success [4]. Some IP's are also

1-4244-0054-6/06/$20.00

available for direct instantiation into SoCs (System on Chip), but so far, no secure circuits make use of reconfigurable logic. One reason for that is the necessity to secure the reconfigurable circuit itself. Access to the reconfigurable area has to be strongly secured in order to prevent this area becoming a perfect backdoor for breaking into the system. Two distinct levels of security are considered. First we have to secure the macro cell against hostile users, i.e. forbid reconfiguration to hackers; in other words, securing the bitstream. The second aspect consists in securing the rest of the system in order to restrict access from this reconfigurable area. It is used to strengthen the previous point and to control interactions of allowed users with in the system: the firewall shown in Figure 1.

Depending on implemented locks, a system will have a lower or higher security level. In [5], authors presented a classification of attackers. A Class I corresponds to clever outsiders. Such attackers often take advantage of an existing weakness in the system. Class II are knowledgeable insiders who have good experience and access to highly sophisticated tools. Class III are funded organizations. They are teams of specialists with great funding resources and have access to highly sophisticated tools.

This paper lists all proposed solutions found in the literature aiming at securing the bitstream. Our focus is to be resistant to Class III attackers.

2. Configuration of stand alone programmable arrays: state of the art.

FPGA can be seen as devices made of two distinct layers: the operating layer and the configuration layer. The purpose of the configuration layer is to store the configuration of the operating layer. The configuration data file is called the bitstream. Additional external non-volatile memory is often used to store the bitstreams and to trigger FPGA configuration. A hostile user may easily probe this configuration bus; this attack is called "man in the middle". Then he can copy this bitstream into other devices of the same family; this is called cloning. The bitstream can be reverse-engineered if this one is not encrypted.

In this section, we will list the different solutions used by FPGA manufacturers to secure the configuration of their devices. In order to embed a reprogrammable array, we will only consider solutions which permit device reconfiguration. We will not consider Fuse / Anti-fuse configuration-based devices.

2.1. Encapsulation

To prevent probing of configuration data, Atmel chose to encapsulate the reconfigurable array and the non-volatile memory into the same package for its FPSLIC family [6]. This solution is efficient against hackers with poor resources, i.e. Class I attackers. But with limited equipment, it is possible to extract the chip from the package, to probe the connexions and to reach the confidential data. Figure 2 is an example of a low cost security, but is not good enough to defeat users of class II and III. Therefore, it is not suitable for our purpose.

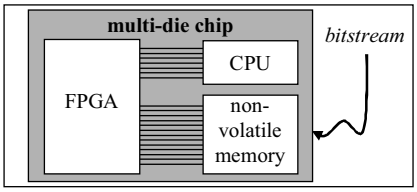

Figure 2. Encapsulation of non-volatile memory.

2.2. Dongle

Another low cost solution consists in connecting the FPGA to a CPLD (Complex Programmable Logic Device) which is used as a "dongle", like software security [7]. The configuration phase is classical. First, the bitstream is stored in a non-volatile memory, and then it is loaded into the FPGA. But during run time, the FPGA sends random data to the CPLD. The security core of the CPLD manipulates this data and returns a result to the FPGA. If the FPGA doesn't receive expected data, it detects that it was cloned and stops running (see Figure 3). This protection is efficient against cloning but not against reverse-engineering. Because the bitstream is not encrypted, it can be analysed, the security core identified and bypassed. Moreover, the cost and the area of a CPLD make this solution impossible for secure circuits like smartcards.

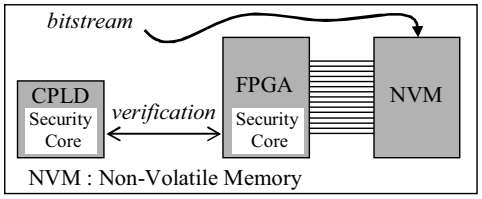

Figure 3. Use of a CPLD.

2.3. Manufacturer defined key

Ciphering the bitstream is a good protection against reverse engineering. A simple way to use cryptography is to store a secret key (in hardware) into the chip when manufacturing it and to store the same key in the CAD tool (Figure 4). Actel used this principle to protect its 60RS family [8]. It implies the implementation of a hard macro into the FPGA to decrypt the bitstream which takes some area, reducing the programmable array for user. This solution is also low cost but is not efficient for our applications because it doesn't prevent cloning. If anybody succeeds in hacking the FPGA or the CAD tool, to obtain the secret key, we can imagine that he will share it on Internet. Then, anybody will have the key and will be able to reverse engineer all chips of this manufacturer. That's

why this solution is not appropriate for embedded secure devices.

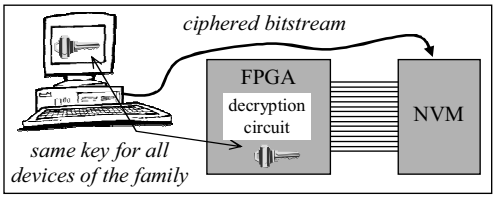

Figure 4. Manufacturer defined key.

2.4. User defined key

An improvement is to allow users to define their own keys. In that way, designs are more protected against cloning and reverse-engineering. The problem is to store this secret key into the FPGA.

In the Virtex-II family, Xilinx [9] uses volatile memory to store the secret key (see Figure 5). This choice was made due to the configuration layer based on SRAM. The cipher algorithm is the triple DES, with 168 bits for the key (in fact, 3 keys of 56 bits). The drawback of volatile memory is that it doesn't keep its content without supply. So it is necessary to add an external battery dedicated to the storage of the key. With the low power consumption of the key storage block, a lithium battery can keep keys for more than 10 years. This solution is robust and has been improved in the last product; Virtex-IV family uses AES algorithm with a 256 bit key and retains data for more than 20 years.

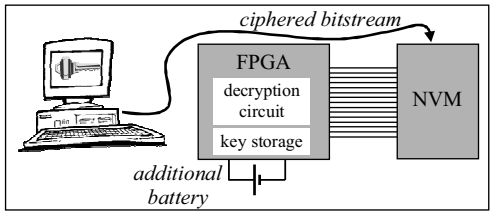

Figure 5. User defined key with volatile key storage.

Let's apply this solution to secure chips. From a security point of view, this can defeat attackers from class I and II, and hinder the work of Class III attackers for a long time. From a technical point of view, we can use a common decryption circuit embedded in secure chips. But depending on the application, additional battery is impossible to use. For instance, it is the case for smartcard area where it is impossible to store bitstream in all card readers, and to implement a battery into the chip. As a conclusion, this solution is suitable for secure components, depending on the domain field. So, this solution will not be considered to develop a standard secure configuration.

One way to free ourselves from external battery is to replace the volatile memory with a non-volatile memory. It's the choice of Altera with its Stratix II family [10]. Its devices use an AES algorithm with a 128 bit key (Figure 6). The principle is the same as previous. In that way, this solution is efficient for our purpose. Moreover, decryption circuit and NVM are present in the secure platform (see Figure 1). The main drawback is the volatility of the configuration layer of the FPGA. This forces us to embed more NVM into the secure chip to store the bitstream which takes more area.

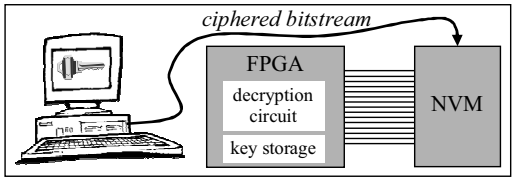

Figure 6. User defined key with non volatile key storage.

2.5. Flash based FPGA

Actel also uses user defined keys in its ProAsic product [11]. The principle is the same as the previous one, except that the configuration layer is made of Flash technology, a non volatile memory. Then, the bitstream is directly stored in the non-volatile configurable array (Figure 7). This small difference has a huge consequence. In its stand-alone devices, Actel doesn't need external NVM. This means that for our embedded array, there's no need of an additional memory, which means a saving in silicon area. The security is efficient and it fits exactly with our platform architecture. This solution is well-suited for our applications, if we can embed sufficient flash memory.

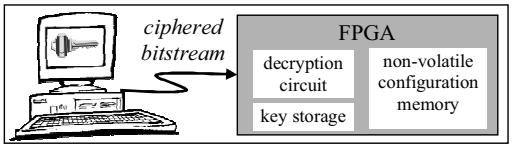

Figure 7. User defined key with non-volatile configurable array.

2.6. T. Kean principle

This principle, proposed by T. Kean for Algotronix in [12], allows the storage of a ciphered bitstream in external NVM without knowledge of the secret key for user or CAD tool. Manufacturers integrate a secret hardware key and a permanent encryption/decryption circuit into the FPGA. During initial programming, the unencrypted bitstream is sent to the circuit via the JTAG port. Then the FPGA uses an embedded key and an encryption circuit to cipher bitstream and to store it in external memory. When power on, the ciphered bitstream from NVM is decrypted by the FPGA thanks to its own secret key (Figure 8). Advantages are that there's no need to know the secret key and the bitstream can only be deciphered by the FPGA that ciphered it.

If we try to apply this method to secure circuits, as used in a bank field, an update is impossible. Because following

this solution, we have to send an unencrypted bitstream to the input port of the smartcard, which is not suitable against "man in the middle" attack.

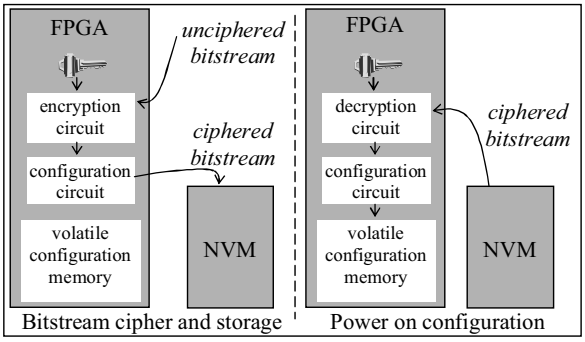

Figure 8. Algotronix principle.

2.7. Dynamic configuration

In [13], the authors improved the last presented technique by using dynamic reconfiguration of FPGA. In order to free some circuit area, they implemented the encryption/decryption circuit into the configurable area. They also allowed ciphering of different blocks of the FPGA with different algorithms depending on the security needed (see Figure 9). This solution seems optimal for stand-alone FPGA because it uses the programmable area to implement only useful blocks. Encryption and decryption blocks are removed when they are not used. From a security point of view, there's a weakness. Bitstreams of the decryption circuits are stored in plain text into the NVM, which means that "man in the middle" can intercept one of these unencrypted bitstreams, and replaces it with a Trojan design. For example, this fake design can copy the secret key to user pad.

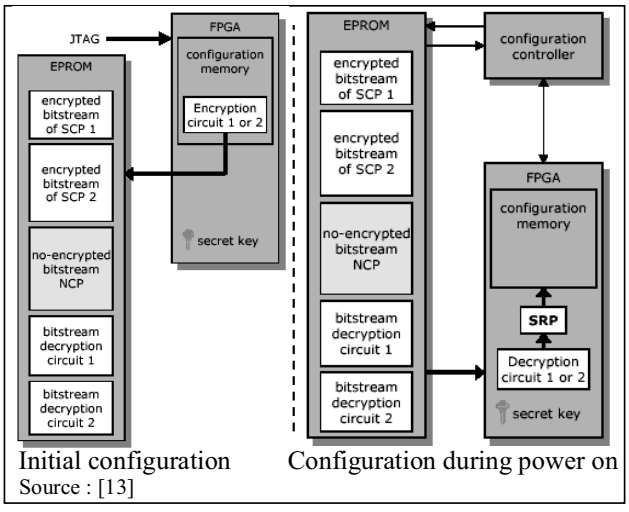

Figure 9. Dynamic configuration.

This weakness is bypassed when used with embedded NVM to store bitstream because the configuration bus is hidden in the circuit. However, it is not suitable for secure circuits because this solution needs a lot of NVM memory to store different bitstreams. That means a lot of area, and an upper cost price. Moreover, like the previous cited solution, there's also a weakness during circuit update. This solution also implies a kind of computation power, not found in all secure circuits.

3. Conclusion

Only two solutions are suitable from a security point of view to be used in secure circuits. Both of them use a user defined key with a non-volatile key storage. The volatility of the configuration layer is the main difference between them. The choice of the appropriate solution will depend on the technology of the embedded configurable array. Flash based solutions seem to be well-adapted, while needing less configuration steps. However, both solutions match. An appropriate architecture for a reconfigurable secure core will be determined in future work.

References

[1] W. Rankl and W. Effing, *Smartcard Handbook – 3rd ed*, Wiley, 2003.

[2] R. Andersen and M. Kuhn, "Tamper Resistance – a Cautionary Note", *USENIX Workshop on Electronic Commerce Proceedings*, Oakland, California, November 18-21, 1996, pp. 1-11.

[3] Hagai Bar-El, "Introduction to Side Channel Attacks", *White Paper*, Discretix website.

[4] "GreenFIELD STW21000 - Reconfigurable Micro Controller", *Technical Article*, STMicroelectronics website, 2005.

[5] D.G. Abraham, G.M. Dolan, G.P. Double, J.V. Stevens, "Transaction Security System", *IBM Systems Journal v30 no 2*, 1991, pp. 206-229.

[6] "AT94S Secure FPSLIC", *Datasheet*, Atmel website.

[7] D. Kessner, "Copy Protection for SRAM based FPGA Designs", *Application Note*, Free IP Project.

[8] "60RS Family SPGA's", *Advanced Datasheet*, Actel website.

[9] "Virtex-II Family Overview", *Datasheet*, Xilinx website.

[10] "Stratix II Device Handbook", *Datasheet*, Altera website.

[11] "ProASIC3 Flash Family FPGAs Datasheet", *Datasheet*, Actel website.

[12] T. Kean, "Secure Configuration of Field Programmable Gate Arrays", *Proceedings of 11th International Conference on Field-Programmable Logic and Applications, FPL 2001*, Belfast, United Kingdom, 2001.

[13] L. Bossuet, G. Gogniat and W. Burleson, "Dynamically Configurable Security for SRAM FPGA Bitstreams", *Proceedings of 11th Reconfigurable Architectures Workshop, RAW 2004*, 26-27 Avril 2004, Santa Fé, USA.