Secure Configuration of Field Programmable Gate Arrays

Tom Kean

Algotronix Consulting, PO Box 23116, Edinburgh EH8 8YB, United Kingdom
tom@algotronix.com

Abstract. Although SRAM programmed Field Programmable Gate Arrays (FPGA's) have come to dominate the industry due to their density and performance advantages over non-volatile technologies they have a serious weakness in that they are vulnerable to piracy and reverse engineering of the user design. This is becoming increasingly important as the size of chips - and hence the value of customer designs - increases. FPGA's are now being used in consumer products where piracy is more common. Further, reconfiguration of FPGA's in the field is becoming increasingly popular particularly in networking applications and it is vital to provide security against malicious parties interfering with equipment functionality through this mechanism.

1 Introduction

In recent years, SRAM programmed FPGA's have established a major competitive presence in market areas previously dominated by mask programmed ASIC technology. For example, SRAM programmed FPGA's such as Xilinx's Spartan family are being promoted for use in consumer products. At the high end FPGA's with a density of several million gates are available. Upgrading of products containing FPGA's by downloading bitstreams in the field is an increasingly attractive option as more and more applications are connected to networks. All of these market trends increase the need for protection of FPGA bitstream information. Consumer products are particularly susceptible to competition from low cost illegal 'cloned' copies. The costs of developing a multi-million gate FPGA design are significant and, therefore, it is desirable to prevent design reverse engineering. Unlike cloning, design reverse engineering for competitive analysis is not, in itself, illegal. In the case of network connected equipment it is important to protect against malicious interference with the download process: maliciously created FPGA configurations can even cause physical damage to the FPGA chip.

The SRAM FPGA bitstream security problem arises because an attacker can probe the connection between the FPGA and the external non-volatile memory during configuration and obtain a copy of the programming bitstream. Many solutions to the problem have been proposed over the last fifteen years [1]. Most have signifi-

cantly reduced the convenience of using FPGA's or had easily exploitable security loopholes and none have been successful commercially. A major selling proposition of antifuse FPGA vendors has been the superior design security offered by their technology [2]. Only within the last few months has a major manufacturer introduced a 'mainstream' SRAM programmed FPGA chip with security features: Xilinx's Virtex II [3].

2 Approaches to Bitstream Security

2.1 Ignorance is Bliss

Until the introduction of Virtex II, the advice from the major SRAM FPGA vendors was that to protect against design piracy by copying bitstream information the best approach was to configure the FPGA before the product left the factory and maintain the configuration in the field using a battery back up when the main power supply to the equipment containing the FPGA was switched off. While theoretically providing a high level of security this was never a practical option for most applications due to the relatively high power consumption of FPGA chips. Battery back up reduces the reliability of the equipment, increases its cost and requires provision for battery replacement in the field.

The conventional approach to preventing design reverse engineering by 'decompiling' the bitstream was for manufacturers to keep the configuration memory layout of the devices secret and only release it under Non-Disclosure Agreement - 'security through obscurity'. FPGA CAD software vendors such as NeoCad nevertheless managed to reverse-engineer FPGA programming bitstreams for the major Xilinx devices. Recently, Xilinx has started to offer Jbits software to support dynamic reconfiguration of mainstream devices [4] which provides an API to bitstream information. With the introduction of Jbits the 'security through obscurity' defence is paper thin and it is only a matter of time before bitstream to EDIF de-compilers become readily available on the internet. At the time of writing there are unconfirmed reports that such software is already in circulation.

2.2 Encapsulation

In October 2000 Atmel announced a secure version of their FPSLIC chip [5]. An FPSLIC is a 'system level integration' device containing an FPGA and a microcontroller. A degree of physical security is achieved by packaging the non-volatile configuration memory with the FPSLIC chip. This constitutes a significant inconvenience to would-be pirates but will not deter a professional adversary. It is relatively easy to remove external packaging around an integrated circuit: this is routinely done by IC vendors to support failure analysis. Once the external packaging is re-

moved it is straightforward to probe the individual IC chips and determine configuration information.

A further downside of this approach to design security is that it restricts the choices for reconfiguring the device. For example, in many systems it may be preferable to share a single large FLASH memory between multiple programmable chips. It may also be desirable to have a larger memory than strictly required in order that multiple configurations can be stored.

2.3 User Defined Key

One of the earliest suggested mechanisms for providing bitstream security to an FPGA is contained in a US patent assigned to Pilkington Microelectronics [6]. The suggestion in this patent is quite straightforward: CAD software encrypts the bitstream prior to storing it in the serial EPROM. The encryption key used is then loaded into a non-volatile key register built from EPROM cells on the FPGA device. When the product is powered up in the field the FPGA can decrypt the encrypted bitstream from the external memory using the key stored in the non-volatile on chip register. This approach protects against both design piracy (since the bitstream stored in the serial EPROM will only configure an FPGA with the correct key stored in its on chip register) and reverse engineering (since the externally available bitstream is encrypted).

The primary disadvantage of this approach is that it requires non-volatile memory within and hence non-standard processing of the FPGA device which increases cost. Copy prevention also requires a different bitstream and encryption key to be generated for each device and therefore complicates the user's manufacturing flow.

A variant of the Pilkington scheme is used by Xilinx in their recently announced Virtex II family. Instead of providing on-chip EPROM memory to store the cryptographic key the key is stored in a key register with its own power supply pins. An external battery maintains the state of this register when the equipment is powered off. Since only the key register is battery backed up very little power is required compared with backing up the entire configuration memory and a small watch battery is sufficient. A key-register only battery backup scheme was independently suggested in a UK patent application filed by Algotronix Ltd. in 1999 [7].

Although the key-register only backup scheme is much preferable to backing up the entire configuration memory it still adds cost to the system and reduces overall reliability. Vibration and shock are of concern because even a momentary loss of power will delete the key. Battery lifespan is affected by self-discharge and lifespan on a printed circuit board may be less than in an environmentally protected location such as inside a watch particularly if humidity is high. In general, provision will

have to be made for service personnel to change batteries in the field. This must be done as preventative maintenance since any loss of battery back up power when the equipment is lost will make the equipment inoperative.

2.4 Secure Serial Memory

Various schemes have been suggested in which a special 'secure' serial EPROM containing encryption circuitry communicates with an FPGA containing decryption circuitry. Such schemes are disclosed in US Patent 5,970,142 assigned to Xilinx [8] and US Patent 5,915,017 assigned to Altera [9]. Schemes based on special secure EPROM's can provide protection against copying the bitstream as it passes between an FPGA and a secure serial EPROM however they have problems with other modes of attack. In so called 'man in the middle' attacks in which an attacker interposes circuitry under her control between the FPGA and the serial EPROM in order to eavesdrop on and make malicious changes to information passing between the FPGA and serial EPROM. In a 'spoofing' attack an attacker designs circuitry to 'impersonate' an FPGA in order to 'spoof' the secure serial EPROM into providing information in a form she can decrypt.

It is possible to design a secure protocol between a secure serial EPROM and an FPGA (although the prior art references cited do not achieve this) but it appears to be a fundamental requirement that the secure EPROM be able to determine that it is communicating with a 'real' FPGA and not 'hostile' circuitry pretending to be an FPGA. The most practical way of the secure serial EPROM confirming the identity of the FPGA is a cryptographic protocol based on secret data known to the FPGA which can be verified by the serial EPROM. Thus, effective schemes based on secure memories also depend on being able to store a secret key in a non-volatile manner on the FPGA.

Since there are alternative methods of providing effective bitstream security using conventional memories given a cryptographic key stored on the FPGA there is little reason to incur the additional inconvenience and expense of schemes based on special secure memories.

2.5 Manufacturer Defined Key

In a scheme supported by Actel in their 60RS family of SRAM programmed FPGA's [10] a fixed key is implanted into the FPGA during manufacture. The method of implanting the key is not specified in the limited documentation available. This fixed key is believed to be the same for all devices and is known to the CAD software which creates a bitstream encoded according to the key. When the FPGA loads the encoded bitstream it decrypts it according to the fixed key prior to storing it in configuration memory.

This scheme provides security against design reverse-engineering but not against 'cloning' since every FPGA has an identical key. Further, there is an 'all the eggs in

one basket' issue in that if the fixed secret key is determined then *all* user designs are affected - thus it is worth an attacker's time to devote considerable effort to determining the secret key. The attacker's task may be made much easier by the fact that key information is embedded in the CAD software as well as the FPGA artwork - decompiling or tracing software is a much easier task than reverse engineering IC artwork.

2.6 Hardware Token Based Schemes

Another possibility which has been suggested is anti-piracy schemes analogous to the 'dongles' or hardware tokens used to protect PC software (for example, the FreeCores proposal [11]). In these schemes a separate chip, normally a CPLD, is provided and connected to user I/O on the FPGA. The FPGA is configured normally from a serial EPROM and the user design on the FPGA then makes contact with the user design on the CPLD. A challenge-response mechanism is provided so that the design on the FPGA can determine that the design on the CPLD is as expected. For example, the FPGA design and the CPLD design might implement identical Linear Feedback Shift Registers, the FPGA then clocks the CPLD a random number of times and compares its output with the output of its own LFSR. If the two do not match the user design on the FPGA determines that it is installed in 'cloned' equipment and disables itself.

The advantage of this scheme is that it requires no hardware support on the FPGA. It relies on the fact that CPLD's are based on non-volatile memory and hence have physical protection against piracy and attempts to extend the protection to a connected SRAM programmed FPGA. However, the scheme provides no additional protection beyond 'security by obscurity' against reverse engineering the bitstream. Dongle based protection schemes for PC software are regularly cracked by decompiling the software binary and disabling the code which accesses the dongle. It would be relatively straightforward to circumvent the piracy protection offered by this approach using a similar technique. In general, it appears impossible to offer strong piracy protection without reverse engineering protection.

A further disadvantage of this scheme is the cost of the external CPLD and the FPGA resources required to implement the security circuitry within the user design.

While this approach makes some sense where piracy is a concern and the FPGA provides no built in security it is insecure and expensive compared with schemes based on encryption circuitry within the FPGA configuration logic.

2.7 Desirable Features in a Bitstream Security System

Based on this description of the prior art some desirable characteristics of FPGA bitstream protection schemes are apparent.

1. The scheme should provide strong protection against both reverse engineering and 'cloning'.
2. No additional components should required on the customer board, so there is no cost penalty.
3. There should be no effect on the reliability of the user board or need for additional service in the field.
4. The user should not have to maintain a database of encryption keys in order to allow for future changes to the design.
5. There should be no significant complication to the manufacturing flow for products containing the FPGA.
6. No changes should be required to the CAD tools or design flow. In particular, no information which could compromise the security of the scheme should be embedded in CAD tools or their supporting files.
7. The scheme should be compatible with standard CMOS processing.
8. The scheme should be based on well understood and standardized cryptographic algorithms and usage modes to allow easy analysis of threats and should not depend on 'security through obscurity'.
9. The scheme should be upward compatible with standard programming modes and standard non-volatile memories. It should allow for design upgrades in the field and design changes during prototyping.

3 A New Bitstream Security System

Figures 1 and 2 illustrate a new FPGA bitstream security scheme proposed by Algotronix [7] which removes most of the difficulties with prior art schemes. The scheme is based on two observations:

1. Flash memory has largely superseded EPROM: thus most modern FPGA's are configured from non-volatile memories which can be programmed in-system by the FPGA itself.
2. If the FPGA is configured within the FPGA customer's facility then there are no security implications to transferring unencrypted bitstreams to the FPGA.

Figure 1, shows the initial configuration of the FPGA within the customer facility. This is achieved during the manufacturing of the board containing the FPGA and involves downloading an unencrypted bitstream via a JTAG interface. The FPGA then encrypts the bitstream based on an on-chip secret key which is unknown even to the FPGA customer and programs it into an external in-system programmable FLASH EPROM. Header bits on the bitstream file are used to indicate its status, for example, insecure bitstream to be converted to a secure bitstream, secure bitstream,

insecure bitstream to be left insecure. With this scheme the FPGA CAD tool flow is not concerned with the encryption process and there is no need for the customer to protect or manage cryptographic keys.

This is only one illustration of how initial programming can be achieved - as with conventional FPGA's it is expected that a variety of modes will be available. Another option would be to pre-program the serial EPROM's with an unencrypted bitstream prior to board assembly and on initial power-on for the FPGA to read in that bitstream, encrypt it and reprogram the serial EPROM with the encrypted bitstream. Similarly, the FPGA may be programmed by a microprocessor on the customer board and return encrypted data to the microprocessor which the microprocessor then uses to overwrite the initial unencrypted data in its memory system.

Figure 2 shows the situation when an FPGA is powered up 'in the field': by examining the header bits it determines that the bitstream is secure and therefore decrypts it using the internally stored key prior to loading it into configuration memory. Although the figure shows encrypted data coming from an adjacent serial EPROM it could equally be provided by a microprocessor or another FPGA in a configuration 'daisy-chain'.

Standard ciphers operating in cipher-block-chaining (CBC) mode can be used to implement the encryption and decryption functions. The choice of cipher is not critical but triple-DES is a reasonable option. CBC mode removes any patterns which would otherwise be present in highly regular data such as FPGA bitstreams, it also provides a cryptographic checksum which can be used to detect tampering with the file. If tampering is detected the FPGA takes appropriate action such as clearing the configuration memory and disables FPGA output pins.

As described so far this security scheme requires a non-volatile key register within the FPGA chip. This could be implemented using a battery backed key register as in Virtex II, however this is not the preferred approach for the reasons outlined above. Instead, it is suggested that laser programmed fuses are used to implant a random key on each chip during the manufacturing process. This is a standard low-cost option at many foundries and has been used in support of redundancy schemes for commodity DRAM chips for many years.

Simplified Piracy Protection

An extension to the scheme outlined above [12] removes the need for laser programmed fuses and provides a high degree of resistance to reverse engineering and piracy while maintaining a completely conventional CMOS flow.

This extension is based on the novel observation that in order to deter piracy it is not necessary to absolutely prevent 'cloning' an FPGA design, only to make it uneco-

nomic to offer a product based on cloned FPGA's. That is, in the vast majority of cases, the problem is that a pirate can compete with the original designer of a piece of equipment by offering 'cloned' equipment at similar or lower cost in the market: not that a pirate could make a small number of units of cloned equipment.

Assume that instead of a non-volatile memory containing the encryption key each FPGA had the same encryption key embedded into the design artwork through changes in one of the masks used in fabrication. The approach of hiding a small amount of secret data in a much larger database is termed steganography and can be a highly secure way of storing an encryption key given the small number of bits involved (less than 200) and the 100's of millions of polygons in an FPGA design database. In fact, arguably, it is harder for a pirate to determine the value of an encryption key when it is hidden in this way than if it is configured into EPROM memory or stored in SRAM. This system provides strong protection against reverse engineering (since the design is encrypted when it is transferred to the FPGA) but no protection against 'cloning' (since every FPGA will successfully load the design).

Based on this analysis prior art systems came to the conclusion that each FPGA chip should have a unique key. However, this is not necessary if the goal is to make 'cloning' uneconomic rather than impossible. Suppose there were five possible keys and FPGA's were supplied with no markings indicating which key was in a particular FPGA. The design owner can use any FPGA since the FPGA will create an encrypted bitstream itself based on whatever internal key it contains. However, a pirate using a naive approach would have to buy, on average, five FPGA's in order to find one which would load a particular pirated design.

More sophisticated pirates might attempt to sort the FPGA's according to their key and resell those they could not use or obtain copies of the design encrypted with all five possible keys and choose the appropriate configuration for a particular FPGA. FPGA manufacturers can counter these more sophisticated schemes in a variety of ways. Firstly, keys can be used in manufacturing for a limited time and then replaced: in this case it may be impossible for a pirate buying 'new' FPGA's from distribution to find one that will accept a design copied from equipment in the field which will contain an FPGA manufactured several months earlier. Secondly, FPGA's can be supplied with different keys in different geographic areas. Thirdly, large customers can be supplied with FPGA's with keys not supplied through 'distribution'. All these approaches make it less likely that a pirate will be able to obtain FPGA's compatible with a cloned design. Lastly, the number of possible FPGA keys can be increased.

Each key variant will involve changes to a particular mask in the FPGA, this represents an additional expense and inconvenience in the manufacturing flow. However, these costs are acceptably small. High volume products will run on multiple fab lines in any case and will, therefore, have multiple mask sets. Also, masks do not last forever and must be replaced from time to time.

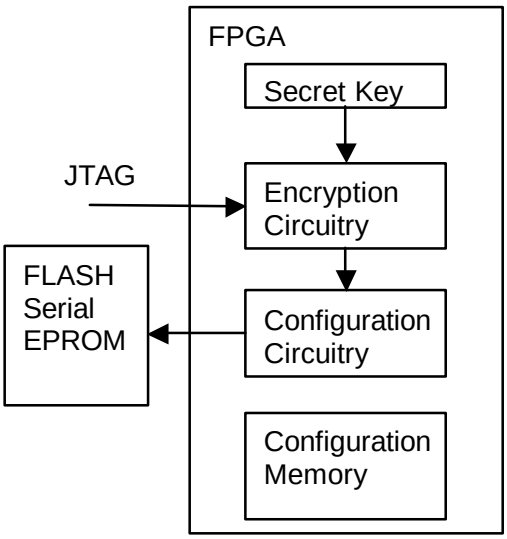

Fig. 1. Initial Programming of Secure FPGA.

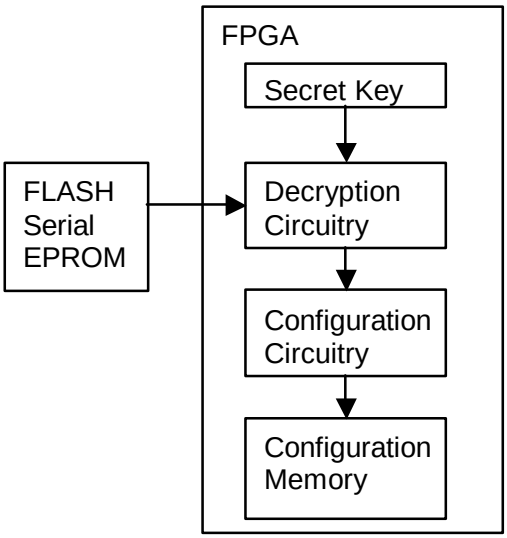

Fig. 2. Normal Configuration of Secure FPGA in the field

6 Summary

Lack of design security has long been the skeleton lurking in the closet of the SRAM FPGA industry. Until recently, customers were willing to live with this problem in order to benefit from the ease of use of programmable logic. Recently, however industry trends have forced manufacturers to address the issue. Provision of strong security technology removes one of the few remaining advantages of antifuse FPGA's and unlocks additional areas of the ASIC marketplace.

Whereas customers may have been willing to ignore security deficiencies when no leading supplier offered a solution now that one, albeit imperfect, system is available bitstream security is likely to become a standard, must-have, feature for all vendors. Similar situations have occurred many times in the past with security technologies: for example, today, all new cars in Europe are sold with engine immobilisers and all e-commerce websites offer Secure Sockets Layer (SSL) security for credit cards.

The proposed security technology offers key advantages compared with alternative schemes: it does not affect system reliability, it does not require additional components, it is compatible with standard CMOS processing, it does not require support from CAD software and it is based on standardised cryptographic protocols.

References

1. Dipert, B., "Cunning Circuits Confound Crooks", EDN Magazine, October 12, 2000.
2. Actel Corporation, "Protecting your Intellectual Property from the Pirates", presentation at DesignCon '98. Available from www.actel.com.
3. Xilinx Inc., "Using Bitstream Encryption", in Chapter 2 of the Virtex II Platform FPGA Handbook available from www.xilinx.com.
4. Steven A. Guccione, Delon Levi and Prasanna Sundararajan, "Jbits: A Java-based Interface for Reconfigurable Computing". Proceedings 2nd Annual Military and Aerospace Applications of Programmable Devices and Technologies Conference (MAPLD).
5. Atmel Corp., "Atmel Introduces Secure FPLSIC", Press Release, Atmel Corp, Oct 12, 2000.
6. Austin, K., US Patent 5,388,157 "Data Security Arrangements for Semiconductor Programmable Devices"
7. Algotronix Ltd., "Method and Apparatus for Secure Configuration of a Field Programmable Gate Array", PCT Patent Application PCT/GB00/04988.
8. Erickson, C., US Patent 5,970,142 "Configuration Stream Encryption"
9. Sang, C., et al, US Patent 5,915,017 "Method and Apparatus for Securing Programming Data of Programmable Logic Device"
10. Actel Corp., "60RS Family SPGA's", Advanced Data Sheet. Available from www.actel.com.
11. Kessner, D., "Copy Protection for SRAM based FPGA Designs", Application Note, Free IP Project, http://www.free-ip.com/copyprotection.html.
12. Algotronix Ltd., "Method of using a Mask Programmed Key to Securely Configure a Field Programmable Gate Array", Unpublished pending patent application.