Secure Configuration of a Field Programmable Gate Array

Tom Kean

Algotronix Ltd., PO Box 23116, Edinburgh EH8 8YB, United Kingdom. E-mail tom@algotronix.com

Abstract Although SRAM programmed FPGA's are generally denser and faster than FPGA's programmed using non-volatile technologies, such as antifuse and Flash EPROM, the need to load on-chip configuration memory on power up makes them vulnerable to piracy and reverse engineering of the user design. This paper discusses an attractive method of addressing this issue.

Introduction As technology and architectural improvements allow re-programmable FPGA's to compete in almost all sectors of the ASIC market and begin to penetrate applications previously addressed by DSP's and microprocessors design security becomes an important consideration [1]. Short product lifespan and competition from nations with less rigorous enforcement of intellectual property rights make design piracy a threat in many consumer applications. With multi-million gate FPGA's being used in advanced applications the economic value of the design programmed into them becomes significant. As FPGA's are increasingly used in critical computational applications, for example in network equipment and in control systems the potential for inadvertent or malicious alteration of the FPGA design becomes significant - particularly in applications involving reconfiguration in the field. For all these reasons the security weaknesses of current SRAM programmed FPGA's are of increasing concern.

Several desirable characteristics for a cryptographic system to protect FPGA bitstreams can be identified: 1) strong protection against both piracy and design reverse engineering, 2) compatibility with standard CMOS processing of the FPGA. 3) no complex key-management requirements 4) compatibility with present programming methods and CAD tools 5) no effect on the FPGA user's product reliability 6) no additional hardware at the board level and minimal FPGA die area overhead

Attacks Several methods of protecting FPGA bitstreams have been suggested in the past but they all have major failings. Most prior art protection methods consider a simple scenario where an attacker monitors the wire transferring programming information from a serial-EPROM to an FPGA (figure 1) and attempts to pirate the design by making a direct copy of the information. Two additional attack scenarios must also be considered: the attacker may use a computer to emulate the FPGA itself or interpose a computer between the serial EPROM and the FPGA monitoring both sides of the transaction and making malicious changes to the signals passing between them in order to subvert the protocol (the 'man in the middle' attack'. Prior-art protection techniques require CAD tools to generate different configuration information for each FPGA or an on-chip EPROM memory [2].

Proposed Solution A secret cryptographic key can be stored on an FPGA in many ways. One method is to use a laser to program a set of links during manufacture. This can be done for a few cents per die and allows each chip to be set up with a different random key.

In-system programmable FLASH memory has largely replaced one-time programmable EPROM storage as FPGA configuration memory. An attractive way of programming a FLASH serial EPROM is to download the desired configuration into the FPGA using the JTAG interface during manufacturing test and have the FPGA itself program the information into the external FLASH configuration memory. Chips such as the Triscend CSoC family [3] which provide an on-chip microprocessor can implement the FLASH programming algorithm in software.

If the FPGA chip also contains encryption circuitry which makes use of its on chip secret key (figure 2) it can encrypt the programming information received through JTAG as it passes through to the serial EPROM. Thus, only encrypted information is stored in the serial EPROM. Since this initial step occurs in the users facility there is no need for cryptographic protection of the programming information as it passes over JTAG.

When the chip powers up in the field it reads encrypted programming information from the serial EPROM and decrypts it using its on chip encryption circuitry and secret key prior to loading it into its configuration memory (figure 3). At this point configuration information is protected cryptographically.

In this security method there is no need for the CAD tools or any person to have knowledge of the particular secret key associated with a given chip. Further: no special processing of the chip is required, the technique does not compromise reliability by requiring a battery to

back up SRAM memory and it is backwards compatible with existing programming methods.

By providing strong cryptographic security to an off-chip non-volatile memory this technique allows for secure storage of any important information which must be preserved after power-down. For example, the off-chip memory can now be safely used to store information such as private cryptographic keys for use in network protocols such as IPSec and SSL. These standard internet security protocols can be used to support secure download of configuration bitstreams from the internet. Without this ability to securely store cryptographic keys an attacker could potentially subvert the download security simply by obtaining access to a product in the field and reading out the keys in its FLASH memory.

Because of the limited space available this paper has only covered the basic principles of this security technique: further details are available from the author.

References

[1] Actel Corporation, "Protecting your Intellectual Property from the Pirates", presentation at DesignCon '98. Available from www.actel.com.

[2] US Patent 5,388,157 "Data Security Arrangements for Semiconductor Programmable Devices"

[3] Triscend Corporation, "Triscend E5 Configurable Processor Family", Product Description, July 1999.

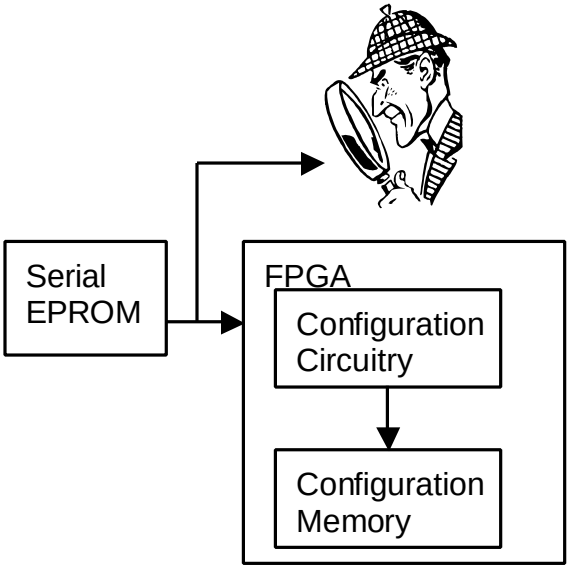

Figure 1. Attacker Monitors the Bitstream

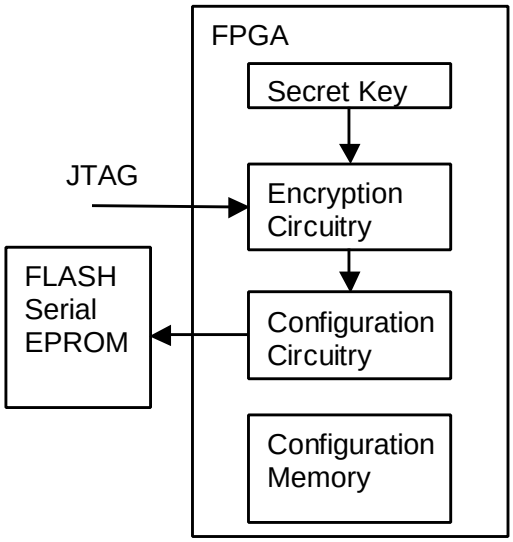

Figure 2. Programming FLASH EPROM during Product Test

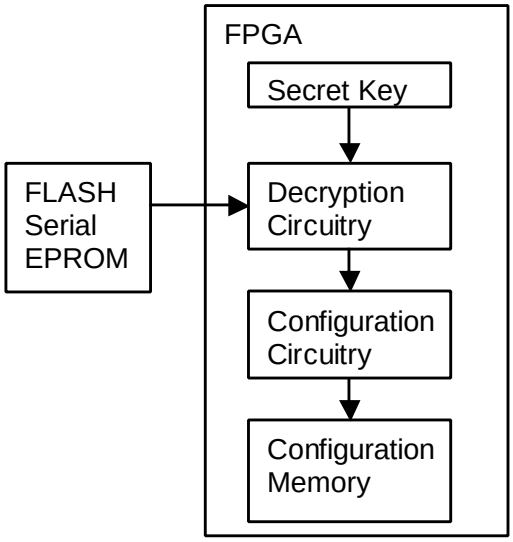

Figure 3. Normal Configuration of FPGA in the field