FPGA Security: From Features to Capabilities to Trusted Systems

Steve Trimberger
Xilinx
2100 Logic Dr.
San Jose, CA 95124 USA
{firstname.lastname@xilinx.com}

Jason Moore
Xilinx
5051 Journal Center Boulevard NE.
Albuquerque, NM 87109 USA
{firstname.lastname@xilinx.com}

ABSTRACT

FPGA devices provide a range of security features which can provide powerful security capabilities. This paper describes many security features included in present-day FPGAs including bitstream authenticated encryption, configuration scrubbing, voltage and temperature sensors and JTAG-intercept. The paper explains the role of these features in providing security capabilities such as privacy, anti-tamper and protection of data handled by the FPGA. The paper concludes with an example of a single-chip cryptographic system, a trusted system built with these components.

Categories and Subject Descriptors
B.7.1. Integrated Circuits, Types and Design Styles. FPGA

General Terms
Design, Security

Keywords
FPGA, Trusted Design, Bitstream Encryption, Cryptography

1. INTRODUCTION

As FPGAs have grown in capability, the value of the applications in the FPGA has grown accordingly. Starting in the early 2000s, SRAM FPGA vendors offered bitstream encryption to protect their customers' bitstreams from reverse-engineering. The usage of FPGAs has continued to grow into applications such as digital cinema, where the data handled by the FPGAs must be protected as well. Further, attacks on the operating FPGA device have grown in sophistication, leading FPGA vendors to provide additional security features. Today, FPGAs provide a large number of features to support secure configuration and operation.

DAC '14, June 01 - 05 2014, San Francisco, CA, USA
Copyright 2014 ACM 978-1-4503-2730-5/14/06…$15.00.
http://dx.doi.org/10.1145/2593069.2602555

2. FPGAS AND THE MANUFACTURING FLOW

The FPGA lifecycle includes two design flows: the base array design and the application design (figure 1), and security must be maintained through both[8]. The base array design is a standard integrated circuit development flow controlled by the FPGA manufacturer. The base array is designed using commercial design tools and libraries, manufactured at a foundry and tested. It is then typically sent to another facility for packaging and final test. The resulting base array is shipped to a customer or authorized distributor. The base array design is subject to all the supply chain trust and security concerns as any other integrated circuit, including questions about tampering with tools, supply-chain control and reverse-engineering. Large FPGA manufacturers maintain a close watch on their supply chain, tracking every device through to final customer delivery or destruction. In addition, they audit their suppliers' systems and processes. As the security issues associated with the design and manufacture of the base array are no different than those of other semiconductor devices, this paper does not focus on the base array design and manufacture, but instead focuses on the security concerns that arise from the need to protect the application design.

The application design also has a design phase, typically performed with FPGA vendors' tools, but often augmented with commercial EDA tools. The application developer integrates design information from a number of sources into an FPGA application: original and re-used HDL code, libraries from the FPGA vendor and other parties and software for soft and hard microprocessors. The FPGA vendor's tools compile the application design into a bitstream, the programming of the FPGA base array to realize the application function. As with any design process, the design itself can be carried out in a secure location, with validated IP and tools. Protection of IP during the design phase is no different for FPGAs than it is for ASICs or microprocessors. Therefore, this paper does not address design-

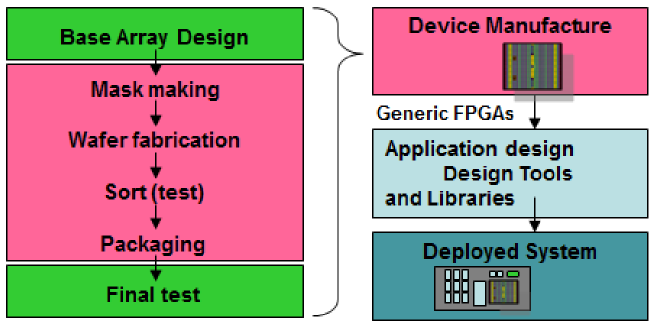

Figure 1. FPGA lifecycle flows. Left: base array. Right: application

phase security.

3. Security in Configuration

A non-volatile FPGA, such as a flash or anti-fuse FPGA, may be programmed before it is shipped. An SRAM FPGA is typically shipped with a separate non-volatile memory containing the programming, and when power is applied, the FPGA loads its programming from the non-volatile memory. This programming step was identified early as a potential security problem.

3.1 Bitstream Encryption

Xilinx introduced bitstream encryption in 2001 in Virtex-II devices address the problem of cloning, unauthorized copy of the bitstream as it is loaded into the FPGA from external memory[6][7]. Since that time, other FPGA vendors have added encrypted-bitstream capability.

Preventing unauthorized copy does not strictly require encryption, since the task from a cryptographic point of view is to determine if the bitstream is authorized to operate in the FPGA. This fundamentally requires authentication, not confidentiality: a device could verify a message authentication code on the bitstream. However, a conceptually-simple attack involves reverse-engineering the bitstream and recompilation[8]. Therefore, reverse-engineering must also be prevented, so confidentiality of the bitstream became a requirement for preventing cloning.

3.2 Bitstream Authentication

Encryption protects only the design, not the data handled by the design. Without some way to deter tampering with an encrypted design, one cannot guarantee that an adversary has not compromised the design to the point where he can extract data from the FPGA. The 32-bit data integrity check on the FPGA bitstream is insufficient to address this attack.

Although there have been no reports of such tampering of FPGAs, Xilinx integrated strong authentication in Virtex-6 devices and 7-series to address concerns of targeted tampering with encrypted bitstreams and the inherent cryptographic weaknesses of a CRC intended only for data integrity[9].

Virtex-6 and subsequent Xilinx FPGAs authenticate using the Secure Hash Algorithm (SHA-256) to compute a 256-bit Keyed Hashed MAC (HMAC)[1][9]. The MAC result cannot be computed without knowing the secret hash key, thereby authenticating the identity of the sender as well as verifying that the message has not been altered. The 256-bit hash size ensures that any tampering with the bitstream will be detected with high probability. HMAC with SHA-256 makes tampering with the bitstream as computationally difficult as guessing the encryption key, which is also 256 bits.

The authentication feature provides resistance to design tampering, which assures the privacy of data inside the FPGA. Privacy of data handled by the FPGA is important in a large number of applications, including digital cinema, network communications and secure database access.

One particularly useful type of data handled by the FPGA is bitstream data. An FPGA with an authenticated encrypted bitstream can reconfigure using the internal configuration access port (ICAP) and still maintain privacy and integrity of design data, basing it on the original bitstream root of trust.

3.3 Configuration Options and Restrictions

Manufacturing tests for SRAM FPGAs require that the configuration data be read back and verified, so this feature is part of the FPGA base array. To prevent theft of the application, readback is disabled when the FPGA is programmed with an encrypted bitstream. Other restrictions include prevention of mixing encrypted and non-encrypted data in a single application, since the non-encrypted application piece might be a Trojan inserted by an adversary. This restriction need only apply to external configuration. A secure application that takes control of its own programming may apply other restrictions on partial configuration, such as restricting the region for the new partial design.

As manufactured, SRAM FPGAs can be programmed with either an encrypted or unencrypted bitstream. Xilinx provides a non-volatile E-fuse that, when programmed, restricts the FPGA to accept only a secured bitstream, preventing a potential adversary from inserting a Trojan design into the system of which the FPGA is a part. Of course, an adversary can still substitute a new, un-programmed FPGA into the system, but this substitution is difficult to carry out in practice.

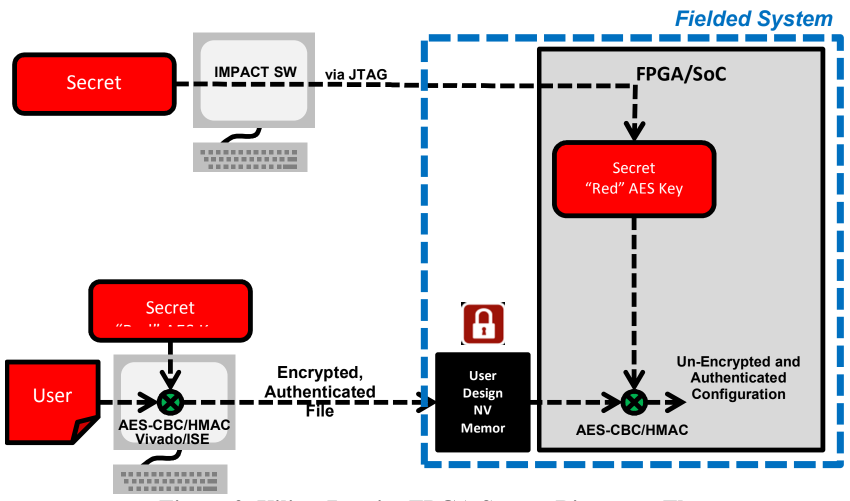

Figure 2. Xilinx 7-series FPGA Secure Bitstream Flows

4. Features for an Operating FPGA

Modern FPGAs include security features available to applications operating inside the FPGA. These features are selected by the FPGA application designer and included in the FPGA application design. They make the FPGA application an active agent in device security.

4.1 Device DNA

Device DNA is a term used by Xilinx to refer a unique identifier for each FPGA manufactured. Device DNA is programmed into the chip during device manufacture by setting one-time-programmable E-fuses. The Device DNA field is typically 56 or 64 bits long, depending on the FPGA family. It is not secret. Anyone can read the device DNA field. The small size and lack of confidentiality of Device DNA preclude its use as a decryption key. Rather, Device DNA may be used to uniquely identify a specific

FPGA device or a range of devices, and restrict the application to function only in those few devices.

4.2 Physically Unclonable Function (PUF)

A Physically Unclonable Function (PUF) is an identifier derived from physical attributes of a specific manufactured device[2]. Like Device DNA, a PUF can uniquely identify a device. A PUF has advantages of privacy and possibly immutability and pamper-resistance. Typically, PUFs are built from FPGA fabric so they can be built of arbitrary size. A PUF may reside anywhere in the FPGA and be unidentifiable by an adversary. However, PUFs are not stable over the lifetime of an integrated circuit. Therefore, to use a PUF as a decryption key, a significant amount of ECC "helper data" is required to ensure a stable key value. The company Intrinsic-ID used a soft PUF structure to uniquely identify FPGAs for their metered-IP solution. [5]

4.3 Bitstream Scrubbing

An adversary may attempt to change individual bits in the FPGA's stored configuration data by focused radiation or power adjustment. Xilinx FPGAs include bitstream "scrubbing" hardware that includes ECC bits for each FPGA configuration data frame. When enabled, scrubbing monitors configuration data and corrects errant bits. Scrubbing has a power cost, so it is not active on all designs.

An application may include an enhancement to the standard scrubbing algorithm by building the scrubbing function using the ICAP to access configuration data. Since the error correction is done in the FPGA application, the application developer selects the number of bits to correct and the encoding of the correction data.

4.4 Program Intercept

As with any complex system, FPGAs include buffers, caches and other temporary data storage locations that aren't explicitly cleared when the device is reprogrammed. This lingering temporary data may divulge sensitive information should an adversary interrupt and re-program the FPGA while it is operating. To address this, the FPGA reprogram signal can be intercepted by the operating application. The application can hold off reprogramming while it clears sensitive temporary data or terminates communications.

4.5 JTAG Intercept

JTAG scan chains are useful in debugging, but problematic for security because they provide access to data and functions throughout the FPGA. In secure applications, an adversary must not have access to the JTAG scan chain. Microsemi and Xilinx provide mechanisms to permanently disable the JTAG interface as well as monitor it internally for activity. Activity on a test port such as a scan chain may indicate an attack in progress. Altera restricts the executable JTAG commands in secured application to a bare minimum.

4.6 Voltage and Temperature Monitors

Xilinx recently added internal monitors on voltage and temperature to its FPGA. These monitors can be used to identify possible environmental attacks on an operating design.

4.7 Key Clear and Device Clear

When an attack is detected internal signals allow the operating application to clear key data or the entire programmed configuration.

5. From Features to Capabilities

Encryption, authentication, Device DNA identifiers, PUFs, bitstream scrubbing, temperature and voltage sensors. These all are features of a security system. But the value is not in the features themselves, but in the security capabilities they provide. These capabilities include prevention of theft of the application design, prevention of tampering with an application before loading, privacy of data handled by the application, both before and during operation and metered IP. Multiple features may be required to provide a capability: depending on the expected attack, authentication alone may not guarantee the privacy of data inside the FPGA. Additional active features may be required.

FPGA bitstream privacy and tamper-resistance provide the basis of further FPGA security capabilities for an application. In the FPGA environment, it is incumbent on the designer of the FPGA application to apply those and other features to achieve security capabilities. The application designer decides whether or not to defend against radiation attacks on the programming of the device. If so, the designer may activate bitstream scrubbing. Similarly, it is the application developer who integrates queries of the on-chip temperature and voltage sensors. Further, the application developer decides what results indicate an attack. Finally, the application developer decides what action to take when an attack is detected.

6. Single-Chip Cryptography

This section gives an overview of a security-sensitive FPGA application called Single-Chip Cryptography (SCC). SCC demands many security capabilities, which are built upon the features discussed in this paper.

SCC combines algorithms and data of different levels of secrecy or control in a single device. The device must not only protect programs during loading, it must also defend against attacks from outside and attacks while operating, including leakage of protected information across internal boundaries.

SCC uses the authenticated encryption capability to load a boot loader. The boot loader manages further FPGA configuration, including software for on-chip processors and data handling. Because it was authenticated and encrypted, the boot loader is known to be unaltered by potential adversaries or accidental bit errors. In addition, sensitive data, such as session keys buried in the boot loader, are known to be kept secret. Authenticated encryption permits trust in the boot loader. That trust can be further applied to additional configuration data handled by the boot loader.

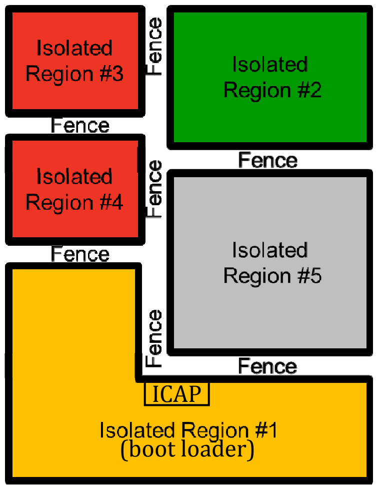

Figure 3: Notional Floorplan of a design into five Isolated Regions

The boot loader accepts further partial device configurations through normal FPGA I/O. It authenticates and decrypts using algorithms in the boot loader itself, constructed using FPGA fabric, rather than using the dedicated FPGA functions. This permits the boot loader greater flexibility in choosing algorithms and in key handling. The boot loader loads various isolated regions through the Xilinx Internal Configuration Access Port (ICAP). The configuration data never leaves the FPGA and if it is authenticated by the FPGA boot loader, it is known to be un-tampered.

To ensure no internal leakage of information between regions, SCC implements the fences of isolation design flow IDF [4]. The basic concept is to take a design and separate critical and/or intentionally separate functions physically on the FPGA. This can be accomplished through careful floorplanning and the use of unused logic as "fences". The empty fence regions are wide enough that a single-bit failure in configuration does not connecting neighboring regions. This separation assures the confidentially of sensitive information even in the presence of accidental or intentional attacks on the fences. Figure 3 shows a block diagram of a design that has been floorplanned with IDF, while figure 4 shows the placed and routed view of the design. Fences are visible as black, unused regions.

In an ideal world, each module would be completely isolated from all others. In practice, some level of communication must exist between isolated regions. Xilinx developed the concept of "Trusted Routing", restricted use of the FPGA interconnects through the fences, such that the isolation established by the use of fences is not compromised.

Loaded as part of the boot loader, bitstream scrubbing, using internal readback, continually monitors the configuration data, in particular the isolation fences, to ensure that changes to the configuration are detected and corrected quickly. SCC can even verify that the Device DNA of the chip, ensuring operation on the proper individual chip.

Without loss of security, the boot loader is itself one of the isolated regions in the device. Any attempt to configure the device from an external source triggers the program signal that is caught internally by the boot loader, which initiates a zeroization of the application inside the FPGA before permitting the re-programming to occur.

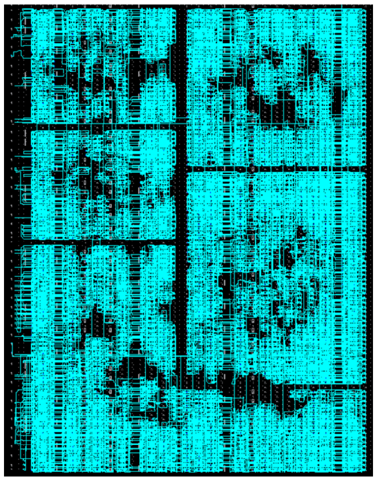

Figure 4: FPGA Editor view of a SCC design with IDF

Originally conceptualized and developed in cooperation with government authorities for FPGAs [3], the application provides additional value in All-Programmable SoCs such as Zynq. Zynq includes both a programmable logic subsystem (PL) that comprises hundreds of thousands of gates of logic, and a processor subsystem (PS) that includes a multi-core ARM processor, caches, memories and peripherals, connected to one another and to the PL using an AXI bus. The Zynq device boots securely, using authenticated encryption capabilities like those described for FPGAs. Zynq provides asymmetric and symmetric authentication, confidentiality and integrity. Leveraging this root-of-trust, applications can implement crypto-processors or systems performing cryptographic functions in the combination of processor and FPGA with confidence that they have not been compromised.

In Zynq, the Processing Subsystem (PS) is known to be isolated from the Programmable Logic (PL). Within the PL, isolated regions ensure separation of sensitive data spatially. Within the PS, known software methods, such as hypervisors and/or ARM Trustzone technology isolate sensitive software processes from other processes. The trusted boot loader decrypts and authenticates all configuration data and software, potentially using session keys and custom algorithms implemented in the FPGA fabric or the ARM processors.

The spectrum of isolation capabilities is suitable to support applications such as the separation of red and black data processing, key management and other high-reliability functions. Partial Reconfiguration is further enhanced. The entire Zynq PL can be reconfigured, or even powered down, controlled by the PS. Alternatively, portions of the PL can be partially reconfigured for applications that require algorithm agility. Decryption and authentication of partial configuration files can be performed by either the PS or PL, allowing users the flexibility to choose their own authentication and decryption algorithms as well as perform functions such as Authenticate before Decryption to aid in defense against side channel attacks.

Starting with the root-of-trust, followed by the power and flexibility of both hardware and software, coupled with the application of isolation technologies and PR, a system that would typically have been developed through the use of multiple devices now could be integrated into just one with no loss of security.

7. REFERENCES

[1] FIPS, "The Keyed-Hash Message Authentication Code (HMAC)", FIPS PUB 198; March 6, 2002, http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf

[2] J. Guajardo, et. al., "Physical Unclonable Functions and Public-Key Crypto for FPGA IP Protection, FPL 2007, IEEE

[3] M. McLean and J. Moore, "FPGA-Based Single Chip Cryptographic Solution," Military Embedded Systems, 2007. http://www.mil-embedded.com/pdfs/NSA.Mar07.pdf.

[4] E. Peterson., "Developing Tamper Resistant Designs with Xilinx Virtex-6 and 7 Series FPGAs," Xilinx Application Note XAPP1084, Xilinx 2012.

[5] Intrinsic-ID, "Quiddikey-Flex," http://www.intrinsic-id.com/products/quiddikey-flex, 2013

[6] A. Telikepalli, "Is Your Design Secure?," Xcell, Xilinx 2003. http://www.xilinx.com/publications/archives/xcell/Xcell47.pdf.

[7] S. Trimberger, "Method and apparatus for protecting proprietary configuration data for programmable logic devices," US Patent 6654889 2003.

[8] S. Trimberger, Trusted Design in FPGAs", Proceedings of the ACM/IEEE Design Automation Conference, 2007.

[9] S. Trimberger, J. Moore, W. Lu, "Authenticated Encryption of FPGA Bitstreams," , FPGA 2011, ACM