Implementing a Secure Boot with Microsemi IGLOO2 FPGA

June 2013

Abstract

Microsemi IGLOO®2 devices have a wide range of differentiated security features that can implement secure boot capability on an embedded system. A secure boot process is needed to verify that the boot code used to 'bring-up' an embedded system is authorized to run on the target processor. Without such a check on the security of the MCU controlled subsystem, a malicious intruder can compromise the entire system. This white paper will educate embedded systems designers of the dangers posed by poor system security and will illustrate how implementing a secure boot, by using Microsemi IGLOO2 devices, can dramatically increase the security of any embedded system that might be subject to outside attacks. Additionally, the paper will demonstrate how, in most cases, the addition of these types of security features can be included for free; since the IGLOO2 FPGA will also be used to implement other common embedded system functions, in addition to the security functions needed for the secure boot.

Introduction

Embedded networked systems control an ever-increasing amount of the modern industrial infrastructure. In the industrial applications arena, embedded systems for smart energy grid installations, complex chemical processing plants, and a host of similar sensitive systems, all require advanced security features to prevent malicious users from system intrusion. Standard approaches to protecting sensitive embedded systems have evolved over the last couple of decades and have created the critical algorithms needed to secure sensitive data and embedded processing functions.

The Microsemi IGLOO2 FPGA family has the world's best suite of complementary security features for implementing protected embedded systems. IGLOO2 FPGAs can implement many security-enhancing capabilities without negatively impacting system requirements. Indeed, because of the innovative technology, holistic security conscious device architecture (from the bottom level silicon process all the way up to the target system), simplified design methodology and extensive support ecosystem, many security capabilities can be implemented without any added system cost, additional power, or increased time to market. Additionally, because in many applications an FPGA is already required to implement a portion of the embedded system, by using an IGLOO2 device to implement these non-security related functions, the dedicated security capabilities can then be utilized to improve system security, at no additional cost. This white paper will use the secure boot as a demonstration of the unique capabilities that enable the IGLOO2 family to deliver increased security for free.

Security in Embedded Systems – A Quick Overview

The security of electronic systems can be divided into two major classes: design security and data security. Design security protects the actual design (intellectual property) associated with the embedded system. Data security protects the data associated with the applications running on the embedded system.

Design security protects the intent of the owner of the design, typically by keeping the design and associated bitstream keys confidential, preventing design changes (insertion of Trojan Horses, for example), and controlling the number of copies made throughout the device life cycle. Design security applies to the device from initial production, includes any updates such as in-the-field upgrades, and can even include decommissioning of the device at the end of its life. Design security should protect against tampering, cloning, overbuilding, reverse engineering, and counterfeiting, as well as providing traceability through the entire lifetime of the system. Some of the important functions needed to implement robust design security include:

- Encrypted key and bitstream loading (using AES-256) to allow configuration to be done in less-trusted locations
- Secure programming with SHA-256 bitstream authentication
- Certificate-of-Conformance to verify correct programming and prevent insider attacks during contract manufacturing
- Supply-chain assurances to eliminate counterfeiting
- Elliptic Curve Cryptography (ECC) for securely loading user keys
- An SRAM-based Physically Unclonable Function (SRAM-PUF) for device authentication

Once the design is secure you can move on to data security. Data security protects the information a device is storing, processing, or communicating in its role in the end application. For example, if the configured design is implementing the key management and encryption portion of a secure military radio, data security could include encrypting and authenticating the radio traffic and protecting the associated application-level cryptographic keys. Some of the important functions required to implement robust data security include:

- Ability to destroy (zeroize) all sensitive stored data in the event of tampering
- User Cryptographic services (such as, AES-128/-256, SHA-256, and HMAC)
- Non-Deterministic Random Bit Generation (NRBG) for secret keys and nonces
- Advanced key storage and management based on a physically unclonable function (PUF), which is a feature of the device analogous to a "biometric"
- Hardware Firewalls to protect sensitive data from unauthorized access
- Advanced anti-tamper techniques, such as Differential Power Analysis countermeasures, to protect secret keys from power side channel attacks

IGLOO2 Architecture Overview

A block diagram showing the key elements of the IGLOO2 architecture is shown in Figure 1 below. The major functional blocks are the High-Performance Memory Subsystem (HPMS), the Multi-Protocol 5G SERDES blocks, the FPGA Fabric, the Multi-Standard GPIO, and the Security Subsystem. In many applications all these functions work together synergistically, but it is useful to first understand the key elements within each block and how they operate. Once this is understood we can look in more detail at how these blocks can work together to create a secure boot implementation in an embedded system.

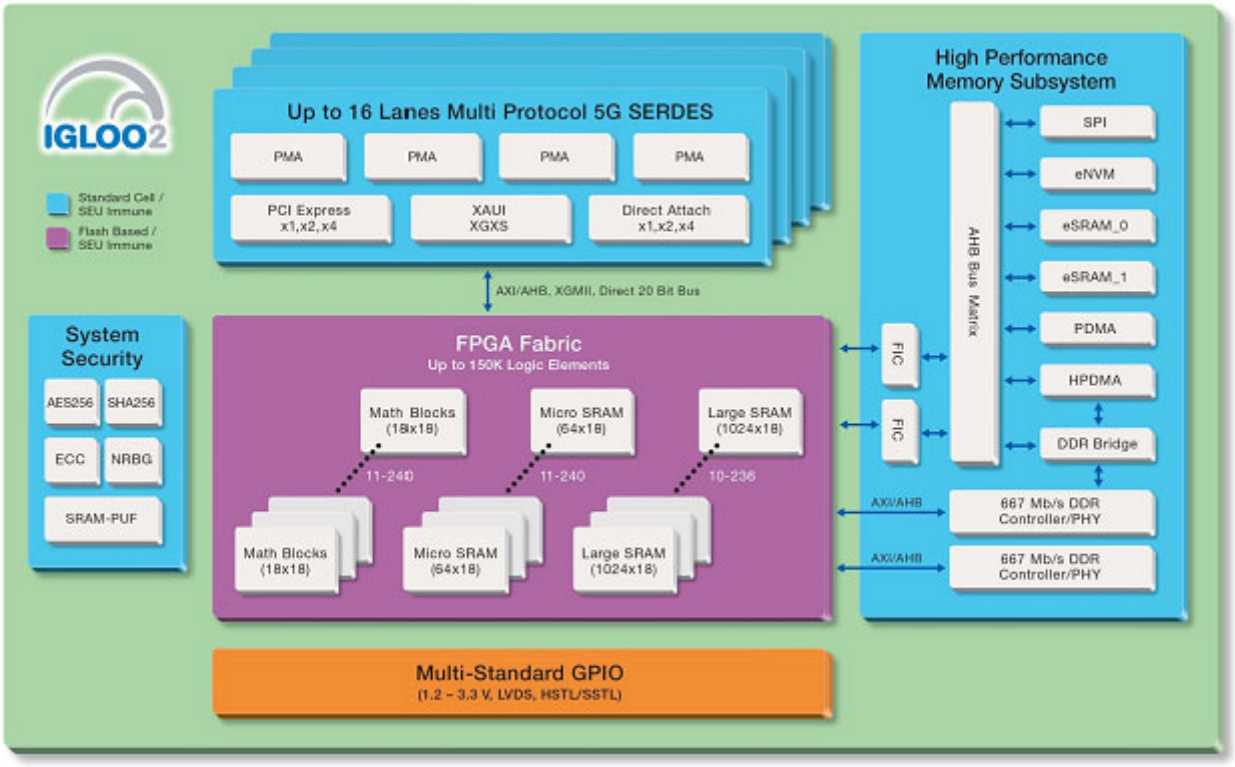

Figure 1: **IGLOO2 Architecture Block Diagram**

FPGA Fabric

The ability to customize user logic and efficiently and easily integrate it with the other key blocks is perhaps one of the most important capabilities of the IGLOO2 family. IGLOO2 devices are optimized for advanced security, high-reliability, and low power consumption applications. Several key features of the embedded FPGA fabric contribute to the best-in-class results the IGLOO2 family delivers. IGLOO2 FPGA fabric is composed of five key building blocks: the logic module, the large SRAM, the micro SRAM, the Math block, and the routing resources that connect everything together. These low-level elements can be used to construct the wide range of functions required in embedded systems designs.

High-Speed Serial Interfaces – SERDES Interface

The IGLOO2 high-speed Serializer Deserializer (SERDES) interface block supports multiple high-speed serial protocols using a SERDES transceiver of up to 5 Gbps; supporting Peripheral Component Interconnect Express (PCI Express®), eXtended Attachment Unit Interface (XAUI), and Serial Gigabit Media Independent Interface (SGMII). In addition, any user defined high-speed serial protocol implemented in the IGLOO2 fabric can access SERDES lanes through the external physical coding sub-layer (EPCS) interface. The SERDES block is configurable to support single or multiple serial protocol modes of operation. The SERDES block is connected to the FPGA fabric through an AXI/AHBL interface or EPCS interface.

IGLOO2 High-Performance Memory Subsystem Description

The IGLOO2 family combines several key memory storage and data transfer functions within the High-Performance Memory Subsystem (HPMS). By combining these functions into a common block, IGLOO2 makes it easier to implement data oriented management, buffering, and transfer functions, like video frame buffers, TCP/IP buffers or PCIe packets without sacrificing the FPGA fabric for these common functions. The 'hardened' blocks of the HPMS results in a more highly integrated, lower power, and lower cost implementation in bridging oriented applications. The descriptions of the key HPMS blocks, as shown in Figure 1 on page 4, are given below.

AHB Bus Matrix: The AHB bus matrix efficiently manages the interconnections between the various blocks within the HPMS.

Fabric Interface Controller: The two Fabric Interface Controllers (FICs) provide connectivity between the FPGA fabric and the HPMS blocks, making it easy to connect the HPMS to application specific functions within the FPGA fabric. The FIC supports ARM® standard AHB or APB busses.

SPI Controller: Serial peripheral interface (SPI) is a synchronous serial data protocol that enables a microprocessor or microcontroller and peripheral devices to communicate with each other. In addition, the SPI controller can interface with large SPI flash and EEPROM devices and includes a hardware-based slave protocol engine.

eNVM: Up to 512 KB of embedded nonvolatile memory (eNVM) is available for code or persistent data storage. An eNVM controller interfaces eNVM blocks to the AHB bus matrix for system access.

eSRAM: The two embedded SRAM blocks (eSRAM) in the HPMS are accessed using an interface controller that provides single-error-correction and dual-error-detection (SECDED) protection. The availability of two large blocks of eSRAM (of up to 64kB), simplifies the implementation of buffers common to many embedded system applications.

DMA Controllers: The Peripheral Direct Memory Access (PDMA) controller manages data transfers from HPMS peripherals-to-memory, memory-to-peripherals, memory-to-memory, and with the FPGA fabric via the two FIC blocks. The High-Performance DMA Controller provides fast data transfer between eSRAM and eNVM blocks and the DDR memory controller associated with the HPMS DDR bridge.

DDR Bridge: The DDR bridge manages multiple AHB bus masters accessing a DDR controller and optimizes the read and write operations to a single external DDR memory by using read and write combining buffers and a round robin arbiter.

DDR Controllers: The two DDR controllers (one accessed via both the HPMS DDR bridge or FPGA fabric, and one just via the FPGA fabric) have an integrated PHY and arbitration logic to simplify the control of external LPDDR1, DDR2, and DDR3 memory devices. The DDR controllers operate at up to 667 Mbps (333.33 MHz DDR) performance and support up to 4 GB of memory.

System Security Block: The IGLOO2 system security block manages all the programming and security related functions included on the IGLOO2 devices. The best-in-class security features of IGLOO2 FPGAs include encrypted user bitstream-key loading, certificate-of-conformance support, X.509 compliant digital certificate management, elliptic curve cryptography (ECC) support, an advanced encryption standard (AES-128/256) engine, a secure hash (SHA-256) engine, a non-deterministic random bit generator (NRBG), differential power analysis (DPA) countermeasures, anti-tampering countermeasures, single use debug passcodes, SRAM-PUF, zeroization, and FlashLock® protection of bitstream decryption keys and security settings. Some of the higher-level functions (like zeroization, AES, SRAM-PUF, NRBG, and SHA-256) are easily included in user designs, as security services accessible via the security block. Many of these features will be described in more detail throughout the rest of this paper.

This overview of the IGLOO2 architecture will be sufficient for our purposes of understanding how to implement a secure boot process in an embedded system but there are many other key elements and additional details are available in the IGLOO2 User Guides on the Microsemi website.

Secure Boot Overview

One of the most important security capabilities to protect embedded systems is a secure boot process. A secure boot process initializes an embedded processing system from rest. It does this by executing trusted code, free from any tampering by a malicious intruder. Without this level of trust, an alternate boot image could replace the original boot code and allow an attacker to ‘hijack’ the entire embedded system. Just about any embedded system needs to be free from such attacks, for many embedded systems such an event could prove catastrophic. Financial transactions could be altered, industrial processes sabotaged, or confidential business data compromised. It is easy to see why security and in particular, the secure boot, is a growing requirement for many embedded systems.

Root-of-Trust – The Starting Point for Implementing Secure Systems

A hardware root-of-trust is essential to system security. It is an entity that can be trusted to always behave in the expected manner. As a system element, it supports verification of system, software, and data integrity and confidentiality, as well as the extension of trust to internal and external entities. The root-of-trust is the foundation upon which all further security layers are created, and it is essential that its keys remain secret and the process it follows is immutable. In embedded systems, the root-of-trust works in conjunction with other system elements to ensure the main processor boots securely using only authorized code, thus extending the trusted zone to the processor and its applications.

The trusted platform module (TPM) is an example of an industry standard root-of-trust. TPM devices provide cryptographic services (hashing, encryption) with a static RSA key embedded in each device. All of the required security features of the TPM are available in IGLOO2 FPGAs. For example, features like on-chip oscillators, cryptographic services, a true random number generator, and stronger design security and anti-tamper measures are all easily implemented in IGLOO2 FPGAs. Additionally, the advanced computational capabilities of IGLOO2 devices (with FPGA fabric, hardened arithmetic functions and block memory) along with the breadth of communications capabilities (including many more I/O pins and many built-in high-speed serial interfaces) make for a vastly superior platform on which to build a robust security system than those provided by typical dedicated TPMs.

Multi-Stage Secure Boot Process Description

Initializing embedded processing systems from rest, requires a secure boot process that executes trusted code, free from malicious content or compromise. Figure 2 below, illustrates the various phases a secure boot process must go through to adequately protect the initialization of an embedded system. Validation of each stage must be performed by the prior successful phase to ensure a 'chain-of-trust' all the way through to the top application layer. The immutable boot loader (Phase-0) code is embedded within the IGLOO2 device and is validated by the secure root-of-trust, which ensures the integrity and authenticity of the code. Each sequential phase of the secure boot is validated by the previously trusted system before code and execution is transferred to it.

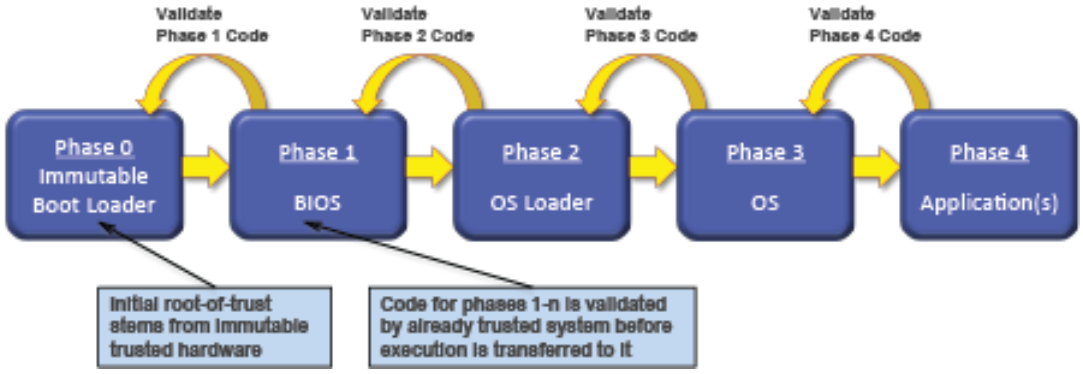

Figure 2: **Typical Multi-Stage Secure Boot Process**

It is essential that any code be validated prior to delivery and execution to ensure that no compromise has occurred that could subvert or damage the boot of each phase. This can be done using either symmetric or asymmetric key cryptographic techniques. One approach is to build an inherently trusted RSA or ECC public key into the immutable Phase-0 boot loader. The developer uses the RSA or ECC private key to digitally sign the Phase-1 code. During Phase-0, the root-of-trust subsystem validates the digital signature of the Phase-1 code, before allowing execution. The boot process is aborted if invalid. It is critically important that the inherently trusted public key and the immutable root-of-trust signature checking process cannot be modified by a would-be hacker. If a hacker could substitute another public key or subvert the process, they could 'spoof' subsequently loaded digitally signed code.

Implementing Advanced Embedded System Security Features for Free

A good illustrative design example makes it easier to identify security requirements and implementation options for networked embedded systems. A pervasive element in these systems is an industrial controller that manages both a high-speed communications interface, such as the Ethernet, that connects the controller to the rest of the installation and a variety of slower speed interfaces for sensors, and process controllers. These types of embedded networked systems are now becoming targets of malicious hackers, as evidenced by the growth in advanced attacks, like the so-called Stuxnet computer worm. A high level of security will be required, with features like the secure boot, anti-tampering, design security, and application level data encryption a given.

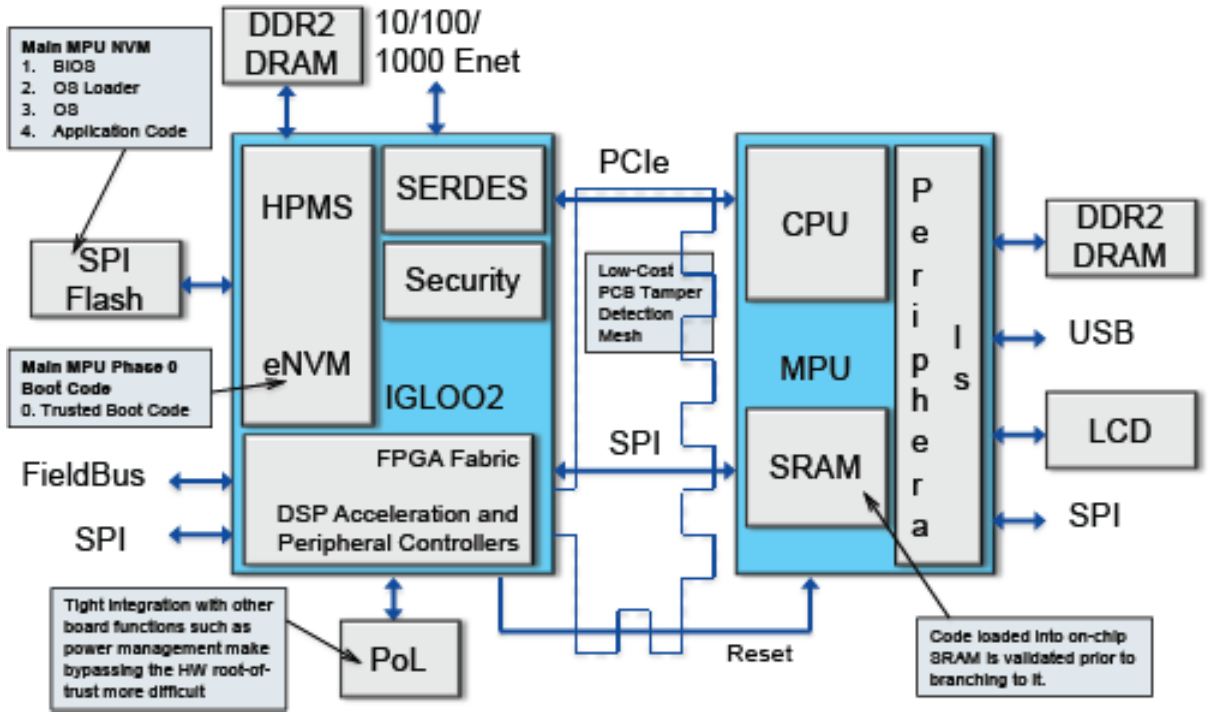

Figure 3: **Industrial Controller with Secure Boot and other Security Features**

Implementing Security Features in the Example Design

Let's now look at how we can implement the security requirements in, for example, the industrial controller, using IGLOO2. A block diagram of the industrial controller is shown in Figure 3 on page 8. The MPU provides the high-level management functions for the industrial controller while the IGLOO2 device provides networking connectivity, algorithm acceleration for computationally intensive sensor operations, such as Doppler computations for velocity measurement and advanced motor control algorithms, to improve motor efficiency and reduce vibrational wear. The DDR2 memory attached to the IGLOO2 FPGA is used for the storage and management of networking traffic, as well as storage for any intermediate data needed for DSP algorithms. A PCIe bus is used as a high-bandwidth connection between the IGLOO2 device and the MPU. The MPU provides a simple user interface via the LCD panel and the SPI port supports a touch screen interface. The DDR2 DRAM connected to the MPU stores image data used with the LCD. A USB port is available as an optional peripheral expansion post.

The IGLOO2 device also serves as the root-of-trust for system security and manages the secure boot process using a multi-stage boot process, as described in Figure 2 on page 7. In our example design, the target processor (on the right side of the figure) is paired with the secure IGLOO2 FPGA (in the middle of the figure), which manages the Phase-0 boot process. The MPU Phase-0 code is stored securely in the IGLOO2 eNVM memory. The IGLOO2 FPGA will manage the Phase-0 boot process ensuring the MPU executes authenticated Phase-0 code. It can independently provide run-time monitoring and apply system penalties, if malicious activity is detected.

In this implementation, all code for Phases 1 and higher, is stored in External SPI flash memory with all code encrypted. During Phase-0, the IGLOO2 device delivers secured code to the MPU to do authenticity checks and decryption of the Phase-1 and higher code. For added security, the Phase-0 code is stored in the eNVM of the IGLOO2 FPGA, which has strong protections against overwriting and could be encrypted while at rest.

After power-up, the IGLOO2 FPGA holds the main MPU in reset until it has completed its own integrity self-tests. When ready, it releases the reset. The MPU is then configured to boot from its SPI port, which is connected to the IGLOO2 SPI port. The IGLOO2 FPGA, acting as an SPI slave, delivers the requested Phase-0 boot code to the MPU as it comes out of reset. Assuming the MPU does not inherently support secure boot, the challenge is to load some code into the MPU, with a high assurance that it hasn't been tampered with.

One approach is to have the very first part of the boot code copy the boot image to the MPU's on-chip SRAM (or cache). Then, the code can perform an integrity check by computing a cryptographic digest of the SRAM contents. This result can be made to vary each time the MPU boots up by including a different true random number, used as a nonce, or 'number used only once,' in the uploaded data. The MPU returns the digest value to the IGLOO2 FPGA for validation. If it does not respond with the correct value, the IGLOO2 FPGA would assume that either the data or the process had been tampered with, and it would terminate the boot process and impose any system penalties enabled by the user design. If everything checks, the boot process would continue by branching to the now-trusted code in the MPU's SRAM. This would contain the code needed to initiate the next phase, and could include a now-trusted RSA or ECC public key.

Once the code in the MPU SRAM is trusted, additional security measures can be employed. This could include establishing a shared key by using public key methods, and encrypting all the subsequent boot code transmitted between the IGLOO2 FPGA and the MPU with that shared key. Additionally, it may be possible to bind all the hardware components of the system together cryptographically so none would work independently unless they are all the exact components of the original system. This makes it much more difficult to attempt a 'divide and conquer' attack strategy typically used in complex systems.

Additional Levels of Security

The requirement to further protect embedded systems from intrusion can also be supported by many of the other advanced features of IGLOO2 devices. For example, the use of flash technology with trusted encrypted user bitstream key loading during device programming, protects the design data against supply chain attacks like cloning, overbuilding, reverse engineering, and counterfeiting. Devices can be programmed by contract manufacturing houses, but the design data can be secured and only the required number of units produced by using IGLOO2 on-chip security keys and encrypted bitstreams. This is a much more secure approach when compared to SRAM-based FPGAs where the bitstream data, data security keys and other sensitive information is located off-chip and must be used to configure the device every time the FPGA is initialized. Design and data security in such systems is problematic.

Tamper Detection and Penalties

Securing the embedded system with the previously described techniques undoubtedly creates a very robust secure environment, but another level of protection can be added to protect against tampering. Hardware tamper detection is one way of determining if the design is under attack. Real-time monitoring of module environmental conditions, such as temperature, voltage, clock frequency and other factors can be used to detect some forms of attack. Many times an attacker will cut traces on the board, in an attempt to isolate devices to implement hardware-based attacks. One simple method of detecting such attacks, shown in Figure 2 on page 7 above, is to use extra I/O signals to route traces throughout the board so that any attempts to cut or drill the board can be easily detected. IGLOO2 devices can easily use FPGA logic to implement a wide variety of tamper detection circuits. If these conditions are detected, the IGLOO2 device can immediately take action and apply a penalty to the system to thwart an attack.

There are several types of penalties the IGLOO2 device can employ to protect the system. Perhaps the most obvious penalty is to reset the system. If the tampering event is determined to be more of a threat, a complete shutdown of the system might be appropriate. This can be easily implemented if the IGLOO2 device controls the PoL power converters during normal operation, as illustrated in our example design in Figure 3 on page 8. Tight integration of the IGLOO2 device with other board functions can make bypassing the hardware root-of-trust much more difficult.

One of the harshest penalties is zeroization. In this case the security system will actually erase (set to zero or one) all the protected data within the embedded system. All MCU code, FPGA bitstream data, security keys, and algorithm implementations will be completely destroyed. IGLOO2 devices have a built-in zeroization capability that can very quickly erase all critical data within the FPGA to completely ‘sterilize’ all of the secure content in an embedded system.

Protecting Security Keys

Once the main MPU is running trusted code, much higher security levels can be achieved with proper design. For many commercial and industrial applications, a secure boot with a few low-cost anti-tamper measures may be enough. For financial and defense applications, additional monitors and a tamper-sensing, tamper-evident enclosure may be required. In all cases however, the security keys must be protected with as many layers of security as possible. IGLOO2 devices have several advanced features that offer the world's best secure key storage, which are not available on other FPGAs.

Some FPGAs permanently program keys within the device while others utilize battery-backed internal SRAM. A superior approach is hardware-based key generation, which creates a device-unique secret key upon power-up. This dynamic key can then be used to form the root-of-trust. Unlike other approaches, the secret key can be transient (ephemeral) and immediately cleared after use, enhancing security since the secret key is never present when the system is at rest.

Variations of individual circuits induced during the manufacturing process can be used to create a physically unclonable function (PUF). The PUF creates a highly secure 'digital fingerprint' of the device using a dedicated SRAM block and a controller.

The controller collects underlying device characteristics resulting in the generation of a unique-per-device hardware-based cryptographic key. IGLOO2 FPGAs employ the Intrinsic-ID® SRAM-PUF along with immutable on-chip embedded non-volatile memory. This and other security features create a root-of-trust for configuration and secure boot of the IGLOO2 device. The IGLOO2 FPGA can extend that trust to securely boot an external processor chip—even if the processor chip has limited or no intrinsic secure boot capability.

Support for Remote Upgrades of MPU Code and FPGA Bitstreams

One advantage of networked embedded systems is the ability to remotely upgrade system features by downloading new code images to the embedded system. The IGLOO2 device can serve as a root-of-trust for any remote upgrade capability using an established application level encrypted data transfer. It might seem like the IGLOO2 device would have difficulty in programming a new bitstream into itself, but because of the dedicated nature of the security functions and tight coupling between the programming functions, IGLOO2 devices can easily support secure remote programming. A special remote upgrade feature also retains a trusted 'golden image' during the update so that a 'back revision' to a known working version is always available.

Protecting Against Side Channel Attacks

Even more sophisticated attacks on design data are evolving and additional features are required to address these new developments. Side Channel Analysis is one of the most sophisticated forms of attack on cryptographic systems that uses information that leaks, unintentionally, from the real-world implementations of cryptographic hardware. For example, an attack might examine the leakage characteristics of a cryptographic device when a variety of encrypted data (also known as. ciphertext) is presented. Even if the encrypted data is random gibberish, measurements and analysis of the power use (called Differential Power Analysis, or DPA), timing responses or electromagnetic radiation given off could provide clues as to the nature of the protected keys or algorithms used within the root-of-trust described earlier. An example of an attack is shown in Figure 4 below, where the stored key can be determined byte-by-byte by observing the electrical current used when different ciphertexts are presented to the device under attack.

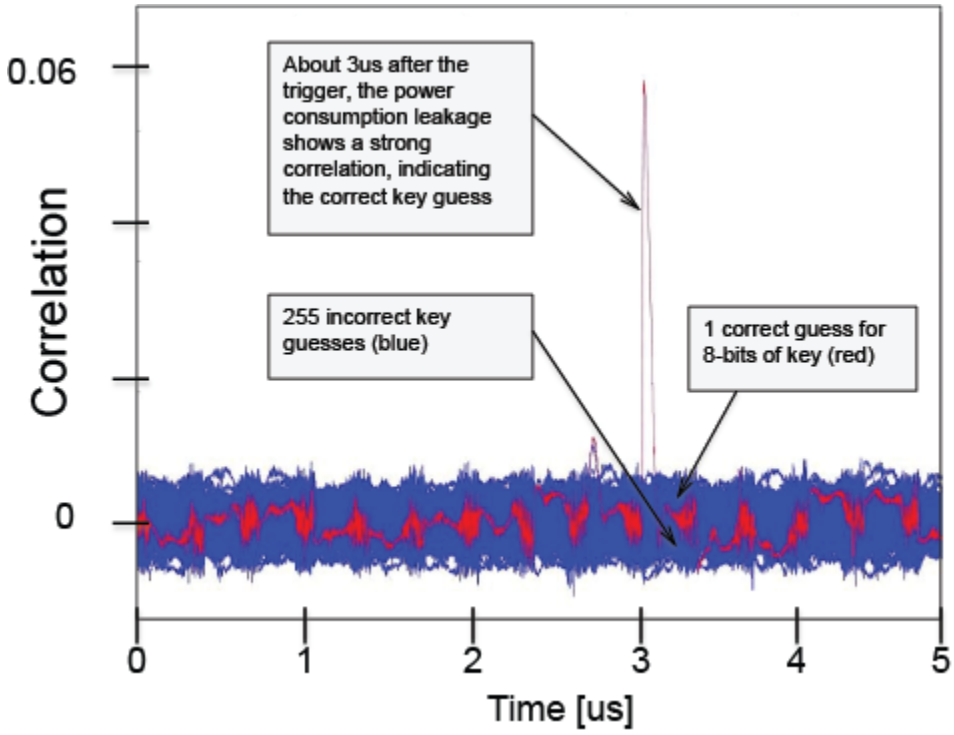

Figure 4: **Example DPA Attack on a Stored Key**

Given enough time and response examples it may be possible to ‘crack’ the stored keys and gain access to the other zones of trust for malicious purposes. IGLOO2 devices include special patented DPA countermeasures, licensed from Cryptography Research, Inc. (CRI, now a division of Rambus), that can protect against side channel attacks to the FPGAs initial configuration, and re-configuration process as well as any active encryption/decryption data processes.

Additional Security Features

IGLOO2 devices have many additional security features that can be used to protect secure content. Here are some examples that can be easily used on the example design. The eNVM can be checked automatically on power up to enhance reliability and security by generating a digest of memory content and comparing it against the expected result so that any natural failures or malicious tampering can be detected prior to a secure boot process. Specialized hardware firewalls can protect the eSRAM, eNVM, and DDR controller from access by a 'blocked' master to restrict activity to known processes. A special JTAG tamper protection feature can be used to restrict JTAG access to eliminate it as a possible attack channel.

Conclusion

The advanced security capabilities of Microsemi IGLOO2 FPGAs can provide the essential root-of-trust and security tool set needed by embedded system developers to meet the challenge of today's security threats. When the IGLOO2 FPGA is also used to implement other embedded system functions – communications bridging, algorithm acceleration, data storage and management, network management, system monitoring and control, or a host of other functions – required security capabilities like secure boot, anti-tamper protection, design security, and data security are all available for free.

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor solutions for: aerospace, defense and security; enterprise and communications; and industrial and alternative energy markets. Products include high-performance, high-reliability analog and RF devices, mixed signal and RF integrated circuits, customizable SoCs, FPGAs, and complete subsystems. Microsemi is headquartered in Aliso Viejo, Calif. Learn more at **www.microsemi.com**.

55900155-0/6.13