Introduction to Implementing Design Security with Microsemi SmartFusion2 and IGLOO2 FPGAs

November 2013

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 advanced over the last couple of decades and have evolved into industry standards that are effective and proven to protect sensitive data and embedded processing functions.

The Microsemi SmartFusion®2 SoC FPGA family has the world's best suite of complementary security features for implementing protected embedded systems. SmartFusion2 SoC FPGAs can implement many security-enhancing capabilities without negatively impacting system requirements. Indeed, because of the innovative technology, security 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 a SmartFusion2 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 secure boot as a demonstration of the unique capabilities that enable the SmartFusion2 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 in a secure platform you can add Data Security functionality. 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 (for example, 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

SmartFusion2 Architecture Overview

The SmartFusion2 family architecture, as illustrated in Figure 1, combines most of the common function blocks required in just about any digital electronics system. The system controller provides overall supervisory control of configuration, power management and JTAG functions. The FPGA fabric includes the user configurable logic and connects to the MSS, DDR controllers, and serial controllers. These main function blocks all work together efficiently to provide the designer with a best-in-class programmable system-on-chip (SoC) for a wide range of applications.

What sets SmartFusion2 devices apart from other programmable devices are the differentiated features that have been included for increased reliability, advanced security, and low power. For example, the sub-blocks shown in purple are immune against single event upset (SEU) occurrences by the use of flash memory cells instead of SRAM cells. These innovations result in dramatic reliability increases compared to SRAM-based programmable devices. Design and Data Security are also radically improved by the inclusion of advanced security functions such as AES-256, SHA-256, SRAM-PUF, and NRBG in the system controller. The next few sections provide an overview of each of the main blocks and their differentiated features that assist in implementing security related functions for your designs.

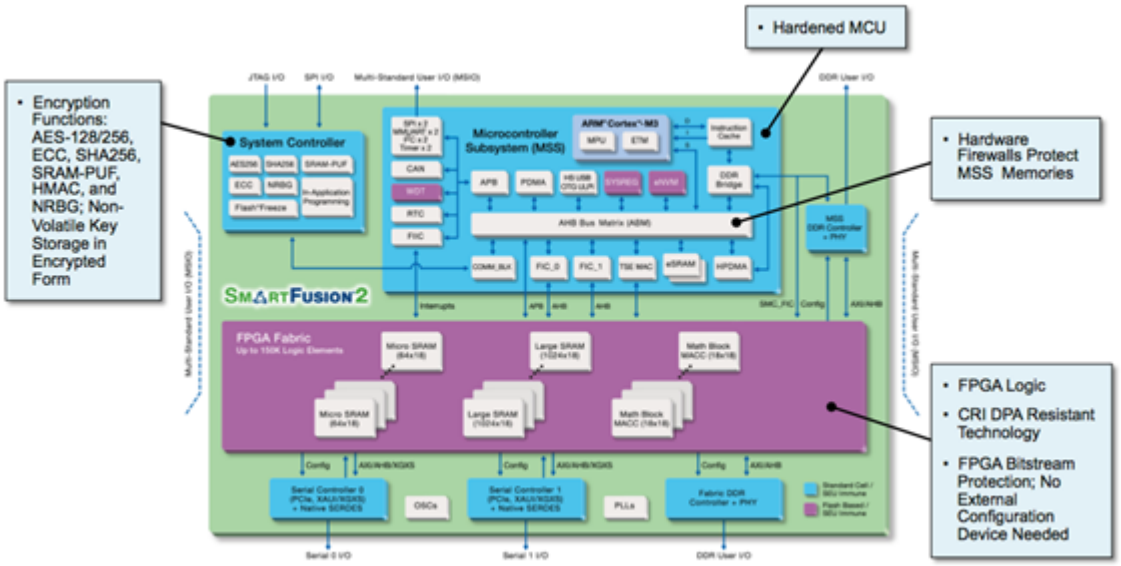

Figure 1: **SmartFusion2 Architecture Block Diagram**

SmartFusion2 Microcontroller Subsystem Description

The SmartFusion2 family combines several key processing, memory storage, and data transfer functions within the Microcontroller Subsystem (MSS). By combining these functions into a common block SmartFusion2 makes it easier to implement control functions using a traditional MCU development process. The 'hardened' blocks of the MSS result in a more highly integrated, lower power, and lower cost implementation for embedded control applications. Key elements of the MSS that can be useful in implementing secure boot applications include the ARM®-Cortex™-M3 CPU; memory related functions including the cache, SRAM, flash code storage blocks, High-Performance DMA (HPDMA) and the external DDR memory controller; a variety of high speed peripherals for data communications including Tri-speed Ethernet, USB, and a peripheral DMA controller (PDMA); several lower speed communications peripherals like SPI, I^2C, UART, and CAN; timers and counters; and interfaces to the FPGA fabric. Of critical importance to all secure boot implementations is the secure storage of the boot-loader code in on-chip eNVM. Only when this 'keystone' segment of the design is secure, can the rest of the boot process be verified and trusted.

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 SmartFusion2 family. SmartFusion2 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 SmartFusion2 family delivers. The SmartFusion2 SoC FPGA fabric is composed of five key building blocks: the logic module, the large SRAM, the micro SRAM, the mathblock, 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 SmartFusion2 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 IGLOO®2 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.

System Security Block

The SmartFusion2 system security block manages all the programming and security related functions included on SmartFusion2 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 SmartFusion2 architecture will be sufficient for our purposes of understanding the key device features and the support available for a wide range of embedded system designs. This paper will now focus on the security features needed to support secure boot capability. Additional details on other SmartFusion2 features are available in the SmartFusion2 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 reset. 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 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. SmartFusion2 SoC FPGAs and IGLOO2 FPGAs have the critical features needed to construct a robust TPM-features like on-chip oscillators, cryptographic services, a true random number generator, and robust Design Security and anti-tamper measures. Additionally, the advanced computational capabilities of SmartFusion2 devices (via FPGA fabric and on-chip CPU) along with the breadth of communications capabilities (more I/O pins and many 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 on page 6 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 SmartFusion2 device and is validated by the secure Root-of-Trust, which ensures the integrity and authenticity of the code.

Each sequential phase of secure boot is validated by the previously trusted system before code and execution is transferred to it.

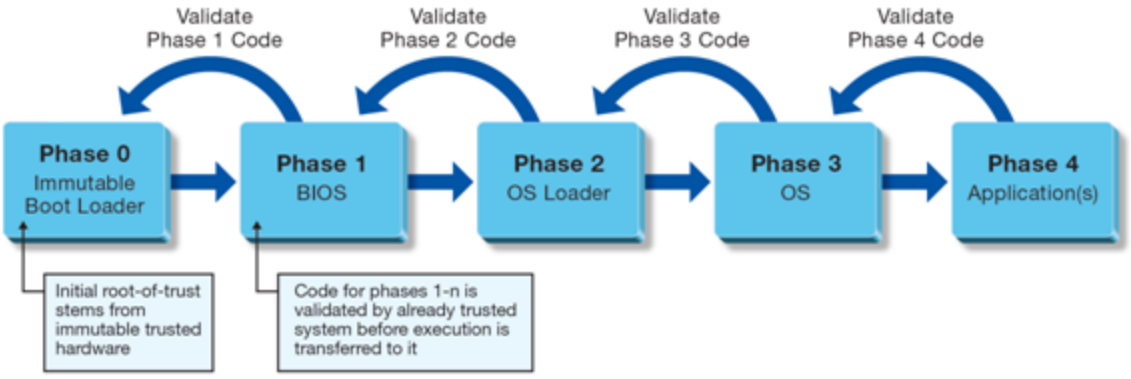

Figure 2: **Multi-Stage Secure Boot Process Overview**

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 next phase of the system boot process. 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.

Industrial Sensor Controller Example Design

A good illustrative design example makes it easier to identify security requirements and implementation options for networked embedded systems. A common element in these systems is an industrial controller that manages both a high-speed communications interface, such as PCI Express®, 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 secure boot, anti-tampering, design security, and application level data encryption a given.

The example industrial sensor controller uses an FPGA and a DSP, as shown in Figure 3 on page 7. The DSP implements the high-level signal processing algorithms while the FPGA provides networking connectivity, algorithm acceleration for computationally intensive sensor operations such as: Doppler computations for velocity measurement, advanced motor control algorithms to improve motor efficiency and reduce vibrational wear and implements the front-end decimation functions. A high-speed Serial RapidIO bus is used to connect the FPGA and the DSP, providing an efficient data transfer capability.

The FPGA also connects to a PCIe bus, used as a chassis management port with remote access via the internet. The PCIe bus can also bridge traffic to and from the RapidIO bus to extend remote management connectivity to the DSP. The external ADC and DAC devices connect to the FPGA using the JESD204x standard, which dramatically reduces the number of signals needed for interconnect, thus simplifying the board layout and minimizing interconnect related dynamic power dissipation. The ADC collects readings from the sensors while the DAC generates analog signals used to control motors and analog peripherals used for sensor positioning and management.

The FPGA controls an external DDR3 DRAM that acts as a buffer for packets to and from the sensor and controller interface, as well as storage for any intermediate data needed for DSP algorithms. This allows the FPGA to offload any low-level data protocol processing and buffer management functions from the DSP. The DDR3 DRAM can also provide storage space for the bridging functions.

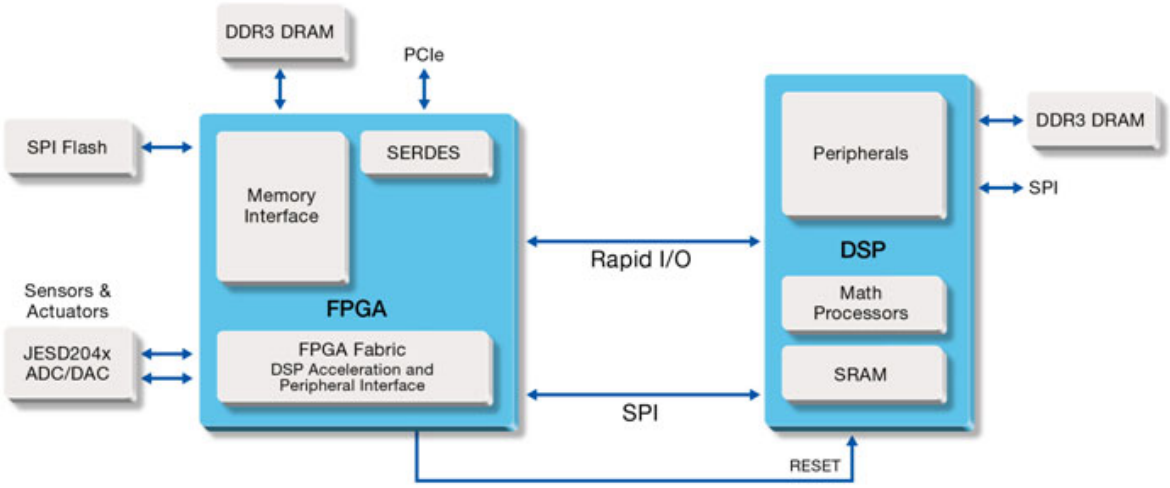

Figure 3: **Industrial Sensor Controller Using an FPGA and a DSP**

The FPGA will also be responsible for 'booting' the DSP from an external SPI flash memory. The FPGA uses the boot function from the DSP's SPI port to mirror the boot process, using its own SPI memory as the DSPs code source. Once the code is transferred, the FPGA allows the DSP to begin execution. An FPGA with a large internal flash block that can be used for the boot code source, like a SmartFusion2 FPGA, means that the SPI memory may not be required.

Implementing Security Features in the Example Design

Let's now look at how we can implement the security requirements in the industrial sensor controller using SmartFusion2 FPGAs, with a focus on secure boot. A block diagram of the controller is shown in Figure 4 on page 8, which illustrates the various elements of the secure boot process. Immutable boot code is stored on-chip along with the security keys. The external SPI memory (in the case where the SmartFusion2 on-chip NVM isn't large enough) stores the balance of the DSP code (including any required OS loader and OS code along with the application code). All of this code is verified using a secure challenge and response system managed by the Root-of-Trust subsystem.

The multi-stage process progresses through several phases, as illustrated inFigure 2 on page 6, with each phase built on the security established at the previous stage.

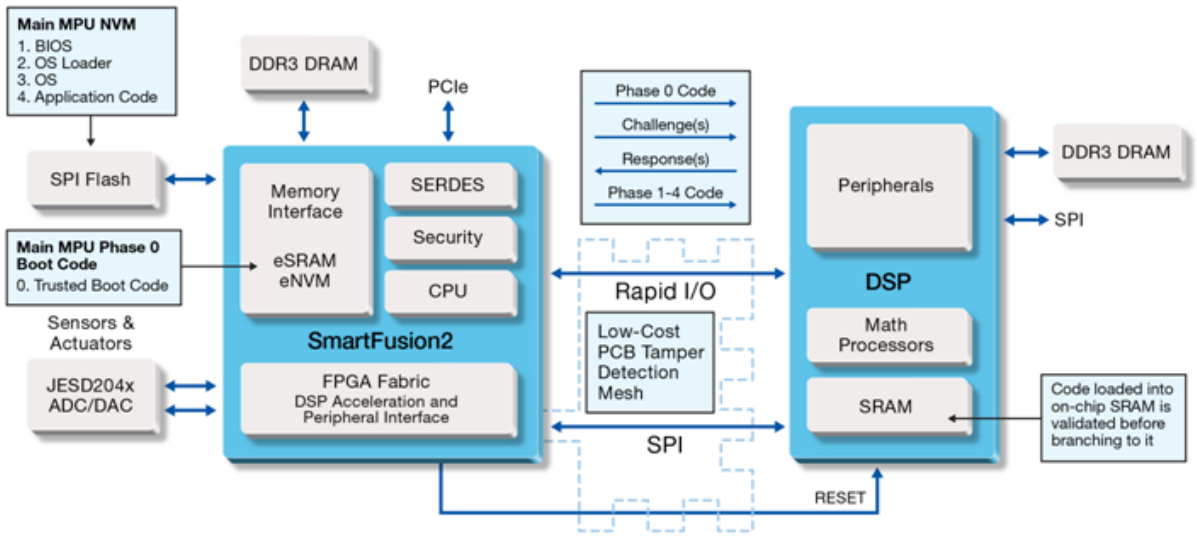

Figure 4: **Secure Embedded Industrial Sensor Controller**

The SmartFusion2 device 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 6. In our example design, the target DSP processor is paired with the secure SmartFusion2 SoC FPGA, which manages the Phase-0 boot process. The DSP Phase-0 code is stored securely in the SmartFusion2 eNVM memory. The SmartFusion2 SoC FPGA will manage the Phase-0 boot process ensuring the DSP executes authenticated Phase-0 code. It can also 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 SmartFusion2 device delivers secured code to the DSP 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 SmartFusion2 SoC FPGA, which has strong protections against overwriting, and could be encrypted while at rest.

After power-up, the SmartFusion2 SoC FPGA holds the main DSP in reset until it has completed its own integrity self-tests. When ready, it releases the reset. The DSP is then configured to boot from its SPI port, which is connected to the SmartFusion2 SPI port. The SmartFusion2 SoC FPGA, acting as an SPI slave, delivers the requested Phase-0 boot code to the DSP as it comes out of reset. Assuming the DSP does not inherently support secure boot, the challenge is to load some code into the DSP, 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 DSP'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 DSP boots up by including a different true random number, used as a nonce, or 'number used only once,' in the uploaded data. The DSP returns the digest value to the SmartFusion2 SoC FPGA for validation. If it does not respond with the correct value, the SmartFusion2 SoC 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 DSP'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 DSP SRAM is trusted, additional security measures for secure boot 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 SmartFusion2 SoC FPGA and the DSP with that shared key. Additionally, it may be possible to bind all the hardware components of the system together cryptographically so no pieces 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 SmartFusion2 devices. These features cover capabilities for advanced bitstream protection, security key protection, secure remote updates, protecting secure application data and tamper protection.

Secure Configuration Bitstream

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 SmartFusion2 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.

Protecting Security Keys

Once the main DSP is running trusted code, much higher security levels can be achieved with proper design. For many commercial and industrial applications, secure boot with a few low-cost anti-tamper measures may be enough. For more secure applications (such as sensitive industrial processing 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. SmartFusion2 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. SmartFusion2 SoC 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 SmartFusion2 device. The SmartFusion2 SoC 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

One advantage of networked embedded systems is the ability to remotely upgrade system features by downloading new code images to the embedded system. The SmartFusion2 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 SmartFusion2 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, SmartFusion2 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. Additionally, advanced security precautions are taken during updates to prevent unauthorized roll-back to previous versions of a configuration bitstream. This is useful in preventing a common form of attack where a previous code revision, with know vulnerabilities or weaknesses, is used as an update to overwrite a newly secure revision.

Protecting Secure Application Data

SmartFusion2 devices have many other security features that can be used to protect secure content. For example, 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.

Tamper Protection

One form of low cost PCB tamper detection is to form a signal mesh using FPGA I/Os so that any attempt to drill holes or cut traces can be easily detected. Various penalties can be applied depending on the security level of the system. A common penalty is to reboot the DSP to see if the problem is only transient in nature. If it persists it may be appropriate to put the DSP into a full reset state and send an error indication to the higher level chassis management system. In the most secure cases it may be appropriate to completely erase all the secure information (keys and code) within the system. This zeroization process is supported natively in SmartFusion2 devices and in the most severe case makes the SmatFusion2 device completely unusable, by removing the ability to be programmed or interrogated.

Implementing Other Operations in the Example Design

Once secure boot and other security related features are functioning, the FPGA can implement the other system requirements. Bridging for the PCIe® bus and RapidIO interfaces can be accomplished using the dedicated PCIe port accessed through the FPGA fabric interface. The RapidIO interface can be implemented in the FPGA fabric and can use the high-speed SERDES in native mode to implement the physical layer interface. Either the fabric DDR controller or the MSS DDR controller can be used in conjunction with the RapidIO and PCIe ports since both have easy access to the FPGA fabric. If transfers to the eSRAM require high performance, the MSS DDR controller with HPDMA assist may be the best option. If traffic will be buffered in FPGA block memories instead, the fabric DDR controller might be the best choice.

The interface to the JESD204x bus, which dramatically reduces the number of signals needed for interconnect, thus simplifying board layout and minimizing interconnect related dynamic power dissipation, is also easily accomplished with the FPGA fabric. The use of several small fabric memory buffers may simplify ADC sensor data collection and the associated preprocessing decimation DSP functions implemented in the FPGA fabric. Similarly, small data buffers may be appropriate for storing data for the DACs used for motor control and sensor related control functions. JESD204B IP cores are available for SmartFusion2 devices (both transmit and receive) to speed implementation.

The pre-processing of the sensor signals is done through the FPGA fabric using DSP blocks, fabric block memories, fabric interconnect, and fabric logic. A variety of filtering algorithms are available in the form of IP cores targeted for SmartFusion2 devices. For example, a Fast Fourier Transform core generator is available to easily create a target FFT function. In particular, a Radix-2 decimation-in-place core is available with rates of 10us for a 256 point implementation.

The SmartFusion2 MSS provides all the capabilities required to manage the overall system. The CPU uses on-chip eNVM and eSRAM for code and data storage to implement the main system management routines. The wide variety of peripherals are available to simplify timing and serial communications tasks. The general purpose timing functions will be useful in managing sensor readings and motor control functions. The peripheral interfaces will help in managing low-speed peripherals used for temperature sensing, fan control, status indications, and other system I/O functions. DMA control can be very effective in improving data transfer efficiency and minimizing CPU overhead and SmartFusion2 devices have two DMA controllers to simplify peripheral DMA and higher-speed memory DMA functions.

The SmartFusion2 CPU can also be used for additional algorithmic tasks to further off-load the DSP so it only needs to run the most computationally demanding functions. In fact, special algorithm acceleration peripherals can be created in the FPGA fabric and can execute complex algorithms with just the management (such as initialization of data buffer addresses, selection of coefficient tables, or other algorithm variable definition tasks) being done by the CPU. A typical architecture could allocate all the ‘simple’ DSP tasks to the FPGA, with the DSP only executing higher level tasks where existing routines are used from the DSP function library to speed development.

Conclusion—Security for Free

A critical point to consider is that since all the non-security functions are already required by the system, the additional security features available in the SmartFusion2 SoC FPGA are virtually free. The secure NVM technology, encrypted bit stream, secure key storage, secure remote update capabilities, tamper protection/penalties, and the multitude of other security related capabilities create the Hardware Root-of-Trust fundamental to all higher level security capabilities. The implementation of secure boot capability, critical to protecting an embedded system from attacks that can threaten significant financial and even potential loss of life, can then be created on top of this secure foundation. Microsemi SmartFusion2 devices are thus the worlds most secure starting point for protecting your valuable embedded system design.

Microsemi has a wealth of information available on its security website. To learn more about security topics related to those explored in this paper, refer to the items listed below or other items available on our website.

To Learn More

Secure Supply Chain

1. Securing Your Supply Chain Life Cycle
2. Secure Architecture in Microsemi FPGAs and SoC FPGAs- An Overview

Secure Programming and Manufacturing

1. Securing Your Supply Chain Life Cycle
2. It’s Easy to Protect Your Embedded System from Theft
3. How Easy is it to Secure Your Designs?
4. What is Design Security in a Mainstream SoC Chalk Talk

Secure System Deployment and Upgrades

1. Introduction to the SmartFusion2 and IGLOO2 Security Model
2. Overview of Secure Boot with Microsemi SmartFusion2 SoC FPGAs
3. SmartFusion2 and IGLOO2 Cryptography Services

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**.

55900167-0/11.13