Microsemi®

It’s Easy to Protect Your Embedded System from Theft

May 2013

Table of Contents

Microsemi

It’s Easy to Protect Your Embedded System from Theft

It’s Easier Than You Think to Protect Your Embedded System Design from Theft

If you have invested years and millions of dollars in the design of an embedded system (and in the creation of the Intellectual Property, or IP, that goes along with the design) it can be of critical importance to protect that system from unauthorized duplication or theft. After all, it’s much easier to steal something as complex as a multi-million gate FPGA design than to create, debug, and test it. The protection of an embedded system that uses FPGAs, is particularly relevant since FPGAs have become the platforms of choice for innovation. Multi-million gate FPGA’s enable the engineer/scientist/entrepreneur to map a new idea to a programmable platform and iterate on that idea until the invention is realized. Because FPGA’s are these platforms for innovation, this defines the IP that is programmed into the FPGA as being valuable. If it’s valuable, then someone will want to steal it.

Design Security is the assurance that the Intellectual Property (the FPGA design) programmed into a device is secure and operates as intended for the life of the product. A majority of the time the primary concern is theft of the IP or the overbuilding of your end equipment, another form of theft. There are cases however, where you need to be assured that the design is not tampered with during the product lifecycle. Take Set Top Boxes (STB’s), for example. STB’s can be provisioned remotely to enable and disable premium channel service to end-users. Clearly, any unauthorized enabling results in a lost revenue stream for the content provider.

Another example, is modifying the injector waveforms in an engine control module. Manufacturers may want to prevent third parties from adjusting the air fuel ratio supplied to engine cylinders, for a variety of reasons. Aggressive fueling of a cylinder can obviously lead to more horsepower and can be quite fun, but may lead to additional engine wear. Without the proper assurance that the ECM hasn’t been tampered with, manufacturers may be making repairs on engines that were tweaked, then flash back to a stock configuration after problems arise. It is true that repair facilities may be able to diagnose the excessive wear via forensics; this, however, takes time and costs money. Why not prevent it in the first place?

These two examples are not “security” applications in the classical sense, as in a Secure Crypto Radio that is used by military personal, but they do have secure aspects to them. Further examples for the need for design security are all around us. We live in a world of mobile computing and machine-to-machine connectivity. Automated vehicles are already here! Robotic or automated home based living assistance is also in our future and it seems like drones are popping up everywhere. Clearly, there is a fair amount of intellectual property in these applications which can and should be secured. Additionally, any application that communicates to others needs to secure any data that flows into, flows out of, or resides within the system (Data Security), perhaps by using standard encryption and decryption techniques.

May 2013

The Keys to Improved System Security

In order to construct a robust security system, it is important to have a way to ‘lock’ the system from unauthorized intrusion and to insure that only valid commands and data are recognized. Typically, security systems utilize special keys; just like the key to the front door of your house to authorize system access. These keys are stored within the system and must also be protected for unauthorized access. For FPGAs there are three main approaches for the container that stores the key.

- SRAM key storage
- One-time programmable fuses
- Non-volatile flash memory
- SRAM Physically Unclonable Function (SRAM-PUF)

SRAM key storage has the characteristic that when power is removed the key disappears. In some applications this can be viewed as a benefit, the primary benefit being the disappearance of the key. Power is typically supplied in the form of a battery, and the need for a battery can carry along some disadvantages. The battery itself increases the material cost, increases printed circuit board (PCB) real estate, and possibly adds regulatory and disposal costs. Batteries also have a finite amount of charge, and you need to replace them every so often, or increase the size of the battery to extend the lifetime of the product. Generally, batteries are a pain to deal with.

One-time programmable (OTP) fuses solve the battery problem but raise other issues. It seems obvious, but you can’t change your keys. The analogy is; if you lost your key to your house and you don’t know who might have it, wouldn’t you want to re-key your lock and get a new key? Losing your house key is synonymous with losing the integrity of your electronic key. Imagine what you’re protecting is not your house (quantity one) but 50,000 units of a fielded electronic system. If lost once (possibly via a hack) access to everything is a major detriment to an OTP key. Ideally, you would like to re-key your locks. Yes, you could have a different key for every electronic device; however, you still cannot change the intended key and all the contents that the key protects could be made available to a rogue actor, essentially bypassing the advantage of having unique keys in each device.

Non-Volatile flash key storage stores the key by using reprogrammable flash memory and offers the convenience of not requiring a battery. If your key is ever compromised, you can simply reprogram the keys in the fielded systems. This approach reduces the system cost over a traditional SRAM-based key storage implementation and dramatically improves peace of mind over the OTP-based key storage designs.

Lastly, SRAM Physically Unclonable Function (SRAM-PUF) is a novel key storage mechanism with superior security attributes. It combines the passive zeroization feature of volatile memory.

Microsemi FPGAs, like the SmartFusion®2 Family, make it easy to implement robust key storage via flash-based key storage and key management facilities that are ‘native’ to the device. The programmer need not spend additional time implementing key storage functions with Microsemi FPGAs; they can be included as part of the overall design and manufacturing flow with minimal overhead for the designer. The variety of key management options available (including SRAM-PUF for the most advanced security requirements) also simplifies the manufacturing, deployment, and end of life processes that can be critical to creating a complete ‘secure product life cycle’ for an embedded system.

Design Security Begins with Your Target Device

So, you’ve decided that Design Security is a good thing. Now, how secure is the device you’re programming your design into? Many devices have advanced cryptographic processing engines or co-processors. Is it enough to say I have an AES (as an example) core on my device; therefore, I have a secure design? The answer is no. The device you’re trusting must be hardened against side channel attacks. Figure 1 shows a classic example of a side channel attack. Here, a safe cracker uses his finger tips to feel tumblers falling while also listening to the same. Through this process he can extract information about the combination using these alternative side channels. This is called a side channel attack. In electronic devices, circuitry performing a cryptographic operation can leak information about its state unknowingly. If the associated circuitry wasn’t designed properly the device can be vulnerable to electronic side channel attacks.

Figure 1: **A Classic Side Channel Attack**

One of the most popular side channels to “listen to” is power supply consumption. Differential Power Analysis (DPA) is pretty straightforward and a quite common method for performing side channel attacks on electronic systems, using data gathered by observing how the power consumption of the electronic system changes during cryptographic processing. With the use of some simple statistical techniques such as, correlation and access to both the cipher text and power consumption information for each cipher text, an adversary can perform DPA and extract the value of the secret key. This is because one of the main assumptions of the ideal mathematical world, a world in which cryptography works almost perfectly, has been violated—the assumption that everything related to the calculations of the algorithm is done in secret (the algorithm uses an un-breakable ‘black box’ to store keys and process cryptographic functions). The result is that the security services are broken; not because of flawed mathematics or improper use of the algorithms or errors in the calculations or protocol errors, but because the implementation of the algorithm in the real physical world is leaking information that can be used to break the system. Due to some pretty fundamental laws of physics and thermodynamics: *in the real world, it is hard to build a black box*. For proper Design Security, not only is mathematical and functional correctness of an algorithm required, but also the implementation must be such that cryptographic operations, keys, or other secrets don’t leak information via side channels.

Paul Kocher and associates, at the Cryptography Research Inc. in the 1990's, discovered that certain cryptographic algorithms leaked enough information via side channels that the secret key could be extracted in one or two uses. When only one or a few measurements are needed to determine the secret, it is called Simple Power Analysis (SPA). When multiple measurements along with statistical analysis are used to extract the secret, it is referred to as DPA. Figure 2 illustrates a side channel analysis setup. Let's assume in the case of that, the input data is an encrypted bitstream for an FPGA. The FPGA can be SRAM-based (on every power-up the bitstream is loaded) or flash-based (bitstream available during programming). Let's also assume that the bitstream is encrypted. The output data is the configuration data for the FPGA. In addition, there is some built-in cryptographic circuitry, typically AES, handling the decryption of the bitstream. Using the setup shown in Figure 2 , the decryption key can be extracted from the power supply side channel using DPA techniques; unless the circuitry decrypting the bitstream and operations involving keys are designed to be resistant to DPA analysis. Once the key is found, the plaintext bitstream can be determined and then reverse engineered. So, not only is it important to have good security algorithms and protocols, but they also need to be implemented with side channel attack prevention in mind.

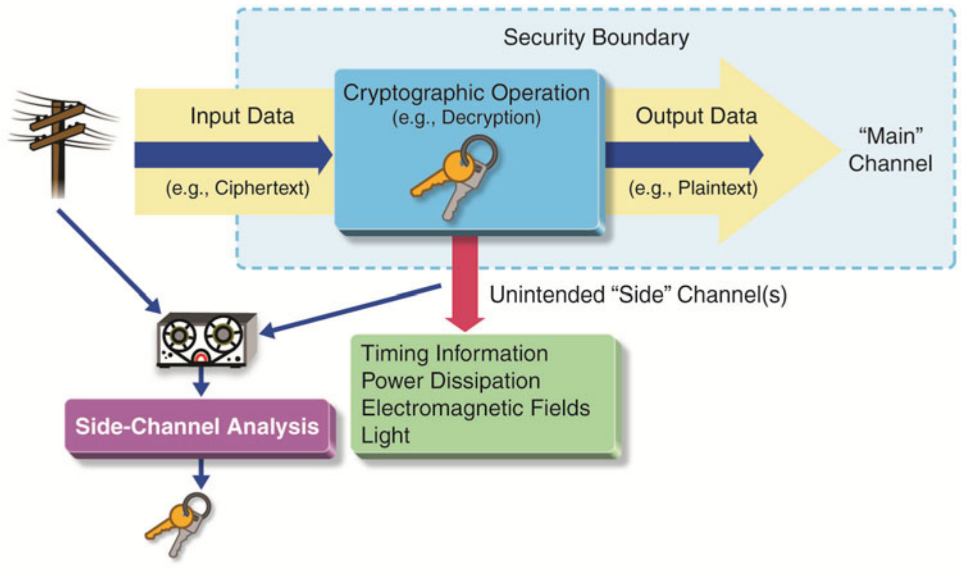

Figure 2: **Differential Power Analysis Side-Channel Attack Methodology Example**

Microsemi is the only FPGA supplier to license the Cryptography Research Incorporated (now Rambus) DPA patent portfolio. Microsemi FPGAs with DPA features, have bitstream and keys that are hardened against DPA analysis. In another first for the FPGA industry, all keys on chip are loaded and stored in an encrypted form. In the unlikely event that the keys were extracted directly from the hardware, the rogue actor would be staring at an encrypted key.

The DPA protection offered by Microsemi FPGAs doesn't require additional design effort from the user. These capabilities create an additional 'layer' of protection around any security keys stored in the device. The designer simply accesses the keys and uses the associate security algorithms, with the knowledge that these actions are resistant to side-channel attacks.

SmartFusion2 Security Model Overview

Figure 3 depicts the simplified security model for SmartFusion®2 SoC FPGAs. There are 4 major components to this model:

1. System Controller
2. Security Segments
3. Secure FPGA Fabric
4. embedded Non-Volatile Memory (eNVM)

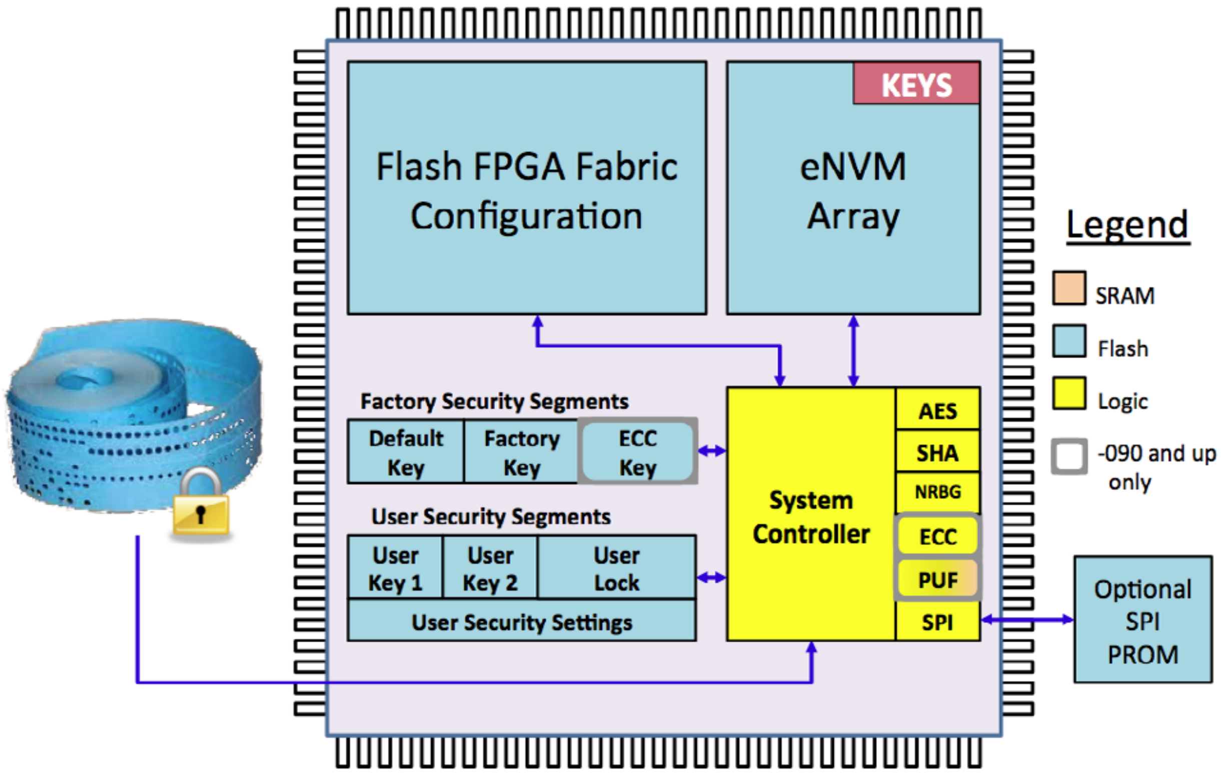

Figure 3: **Simplified Security Model for SmartFusion2**

System Controller

The SmartFusion2 System Controller manages all programming, verification, design security key-management, and related operations. The System Controller is a dedicated fixed-function hardened processor reserved for these functions, and is not reconfigurable in normal operation. Its programming and runtime operation are determined by a dedicated immutable metal-mask ROM. It also includes some dedicated scratch-pad SRAM, and hardware accelerators for specific cryptographic function support.

During programming, the system controller authenticates and decrypts incoming bitstreams, erases and writes the target flash memory segments, and responds to other external programming-related protocols, such as key verification. The System Controller essentially programs what the bitstream tells it to in a secure, authenticated fashion. If the bitstream contains only an eNVM image then that's what gets programmed. Possible bitstream configurations are:

- FPGA fabric only
- Entire eNVM only
- Portion of an eNVM
- Security keys and settings
- Combinations of the above

During normal runtime operation, the System Controller boots the FPGA and ARM®Cortex™-M3 CPU and provides optional cryptographic services for advanced data security users. These services can include functions, such as an Advanced Encryption Standard engine, a SHA-256 Secure Hash engine, a Non-deterministic Random Bit Generator (NRBG), Elliptic Curve Cryptography, and SRAM-PUF.

Security Segments

The heart of the security system is the robust storage of security keys. SmartFusion2 devices have several features that make it easy to create a robust key storage function. There are three user security segments ("User Keys 1", "User Keys 2", and "User Lock"). The first holds a user bitstream encryption key (UEK1). It can be the root key for encrypting and decrypting bitstreams, and for authentication of bitstreams. This segment also holds the FlashLock® passcode. It is used to unlock access to the three user security segments, if re-entered and matched correctly. A correct match allows the user to update the segment contents. It also unlocks many of the user lock-bits, until such time as the device is reset; allowing operations such as programming or verification to continue that may have been disallowed by the stored lock-bit settings.

The second user keys security segment stores a second user encryption key (UEK2). This can be used interchangeably with UEK1. This segment has its own passcode, which allows overwriting of the key and passcode, after the passcode is successfully matched (and before the device is reset). This passcode does not unlock anything in the User Key 1 or User Lock segments. Use of the second user keys segment is strictly optional. Having a second user key can facilitate use models that would be difficult to implement with just one user key, such as using the secondary key to program or update a subset or class of products versus all products that may be in the field.

The User Lock segment holds the majority of lock-bits for setting user security options.These include options for configuring both design and data security. Many of the lock-bits are overridden if the FlashLock passcode is matched (assuming that other security options allow the FlashLock passcode to be active).

The FlashLock passcode is programmed as part of the initial user key injection procedure, and is stored in the User Key security segment in hashed form. If it is re-entered and matched later, it will temporarily unlock the three user segments, allowing changes to the keys, passcodes, and security settings (i.e., lock-bits) stored there. The unlocked state returns to the normal locked state (as defined by the lock-bit settings) on the next device or JTAG reset, or power cycle.This is also when any permanent changes written to the security NVM segments will take effect.

Finally, the Permanent FlashLock mode can be used to turn a SmartFusion2 device into an OTP device. This mode is considered quite secure because it disables most programming, verification, and debug operations. There are additional optional, layered, security settings (i.e., additional lock-bits) that when used in conjunction with Permanent FlashLock mode can provide even higher levels of security:

- The factory test mode can be permanently disabled.
- The programming ports can be partially disabled:
  - The JTAG boundary scan instructions can be disabled (EXTEST, SAMPLE/PRELOAD, CLAMP, HIGHZ, and EXTEST2).
  - Virtually all JTAG and SPI programming instructions can be disabled so they are ignored when parsed, even before other lock-bits (e.g., for disallowing overwriting of the NVM) are checked.

Secure FPGA Fabric

Microsemi flash-based FPGA configuration memory cells are located within the FPGA fabric and directly control the routing switches and look-up tables used to implement user logic. This means there is no turn-on delay due to moving configuration data from nonvolatile memory to control these resources (instant-on). Security is also greatly enhanced since configuration data is not stored off-chip and therefore, does not need to be loaded and decrypted on each power-up, exposing the associated keys to side channel analysis during their use. Even though SmartFusion2 devices only need to be configured once during manufacturing, or very infrequent field updates, as an added layer of security (to prevent reverse engineering of bitstream configuration files) SmartFusion2 devices only accept encrypted bitstreams. Researchers claim, and history confirms, that plaintext bitstreams can and will be reverse engineered. Allowing only encrypted bitstreams makes it more difficult for the bitstream format to be reverse engineered by researchers, malicious users, or adversaries.

Protected Storage in the eNVM Storage Array

The embedded Non-Volatile Memory (eNVM) storage array within SmartFusion2 devices provides additional security capabilities to the designer. This memory array can be organized as separate pages with pages configured for specific functions. For example, pages can be designated as write-protected to make it easy to control sensitive data. Additionally, a novel NVM integrity check mechanism can be used to check the reliability and security of a device automatically upon power-up, or upon demand. The contents of all the nonvolatile configuration memory segments, FPGA fabric configuration, plus any eNVM pages declared as write-protected are digested (hashed) using the SHA-256 algorithm. The results are compared to values stored in dedicated nonvolatile memory words located in each segment. If the contents are unchanged from when the digests were computed and stored during the original programming steps—that is, if the current and stored digests match—the test will pass; otherwise a failure is flagged. This test provides assurance against both natural and maliciously induced failures.

These security features make it easy to provide higher levels of security without additional design effort or the use of additional user logic.

Conclusion

The world is on an inexorable march towards open communications and automation.This is not new; this trend has been going on since time immemorial. What is new is that the pace of innovation is accelerating. The need for Design Security is now more important than ever before. Microsemi FPGAs are the only FPGAs on the market that have been designed with security in mind and implement key features that dramatically simplify the creation of secure embedded systems.The use of on-chip flash-based secure configuration memory, the inclusion of advanced features for robust secure key storage, DPA resistant hardware, and a variety of security services and options to protect design IP over the entire system life cycle dramatically simplify the creation of secure embedded systems. Microsemi FPGAs make it easy to provide a level of protection for your designs that no others can match.

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

55900152-0/6.13