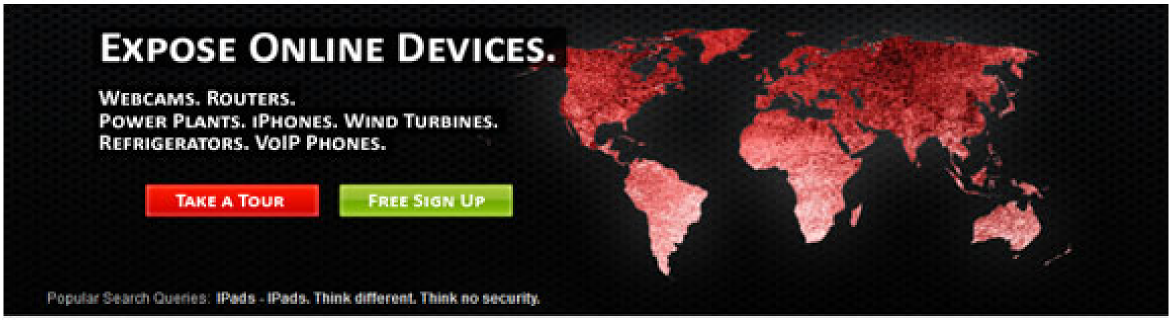
Design of a Secure Wireless Communication System

Application Example

Introduction

The dramatic growth in network connected embedded systems has been driven, at least in part, by the ability to support remote management functions, eliminating the need for in-person inspection and system updates. Unfortunately, remote access of an embedded system can also invite malicious intrusion, if proper security features are not included in the design. The Shodan computer search engine home page, a portion of which is shown in Figure 1, identifies several types of connected devices that can be searched for and identified. When combined with network testing software like Metasploit, that can be used to penetrate a network (an illegal use of course), any unprotected system can be put at risk. Frighteningly, a recent research paper (refer to the "References" section at the end of this paper) showed that using Shodan uncovers over 500 million internet control systems that could be at risk.



EXPOSE ONLINE DEVICES.

WEBCAMS. ROUTERS.
POWER PLANTS. IPHONES. WIND TURBINES.
REFRIGERATORS. VOIP PHONES.

Popular Search Queries: IPads - IPads. Think different. Think no security.

Figure 1: **Shodan Search Engine for Finding Network Connected Devices**

Digital Signal Processor (DSP) based designs can be particularly vulnerable to intrusion when the DSP doesn't have sufficient native security capabilities. In many applications advanced security capabilities can be implemented for free when a companion FPGA is used to offload the processor. Furthermore, if the companion FPGA uses a technology that is inherently secure from copying or cloning you can also protect your design from these types of theft. Let’s look at an example design in more detail to see how this can be accomplished.

Wireless Communications System

In a wireless communications system example that uses an FPGA and a DSP, as shown in Figure 2 on page 2, the DSP implements the high-level signal processing algorithms while the FPGA 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 through 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 board layout and minimizing interconnect related dynamic power dissipation. The FPGA controls an external DDR3 DRAM that acts as a buffer for packets to and from the wireless interface and allows the FPGA to also 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.

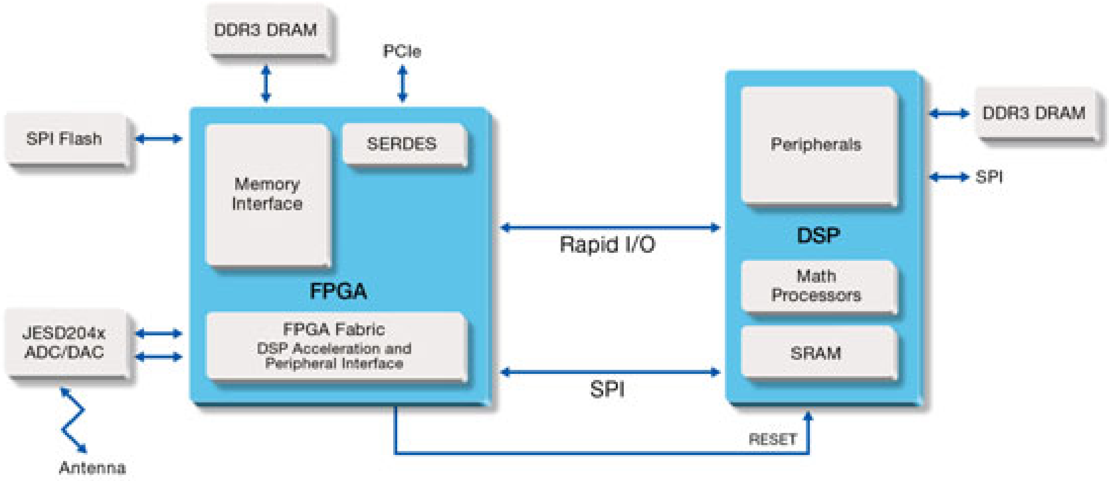


Figure 2: **Wireless Communications System Using an FPGA and DSP**

The FPGA will also be responsible for ‘booting’ the DSP from an external SPI flash memory. The FPGA uses the boot function from the DSPs 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. If your FPGA has an internal flash block that can be used for the boot code source the SPI memory may not be needed.

Security Requirements

A troubling trend in embedded systems is the growth in attacks against embedded systems with connections to the internet, or to any remote entity. When these systems have remote update capabilities, they can be even more vulnerable to attacks. In particular, if systems don't protect the boot processes an intruder can substitute their own code and effectively hijack the entire system. This can result in damage to the system, significant financial losses, and possibly personal liability. A secure boot process needs to be used to minimize such attacks and a Hardware Root-of-Trust (RoT) is required to implement any secure process.

A Hardware RoT supports verification of system data integrity and confidentiality, as well as the extension of trust to internal and external entities. A Hardware Root-of-Trust is known to be secure from intrusion or modification and can thus be the starting point for proving that higher-level functions are also secure. In embedded systems, the RoT 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 Hardware Root-of-Trust needs to build on a secure FPGA, since only a secure FPGA can be trusted.

This means that the FPGA configuration bitstream must be protected from copying or reverse engineering so a malicious intruder can't break the root-of-trust. Another way to say this, is that securing the Intellectual Property (IP) of the FPGA is a necessary requirement for securing the rest of the embedded system.

Security Requirements for a Multi-Stage Boot Process

Initializing embedded processing systems from rest requires a secure boot process that executes trusted code, free from malicious content or compromise. Figure 3 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 can be embedded within an FPGA device and is validated by the secure RoT, using protected security keys and associated security algorithms, to ensure 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. In the wireless communications example design a smaller number of stages may be required, but the fundamental methodology is the same**.**

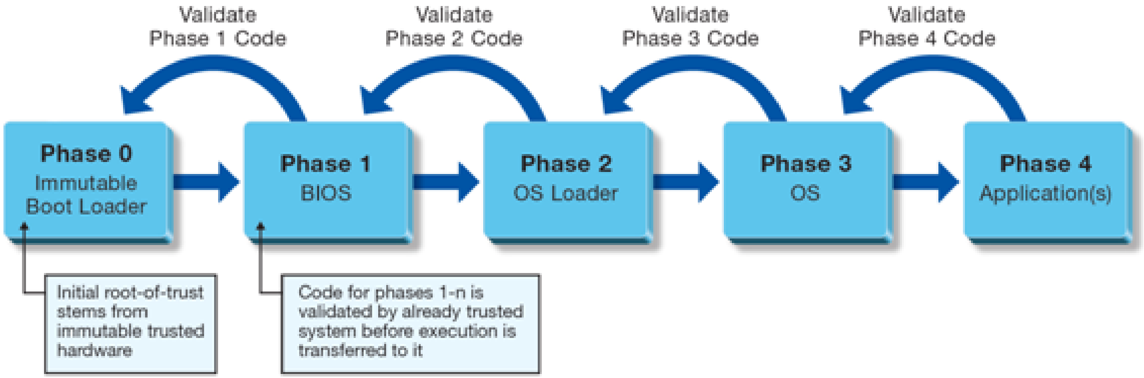


Figure 3: **Multi-Stage Secure Boot Process Overview**

Implementing a Secure Wireless Embedded System Using Microsemi FPGAs and SoC FPGAs

Now that the key security requirements of IP protection, Hardware Root-of-Trust and secure boot have been described, let's look at an example implementation showing how these requirements can be satisfied. First, let's look at how to protect the IP for our FPGA by protecting the configuration bitstream.

Design IP Protection

A common method of protecting configuration bitstreams that are used on power-up (like those used for SRAM-based FPGAs) is to encrypt the bitstream data. This makes it more difficult to capture the bitstream by just observing it during the configuration start-up process. A decryption key is stored within the FPGA and is used to decrypt the data prior to configuring the FPGA fabric. The key must be retained when power is removed, so a battery is typically required to retain the security key.

Another approach to securing the FPGA configuration bitstream is to store it completely on-chip using non-volatile memory. Because the configuration bitstream isn't exposed during start-up, it is much more secure. Some FPGA's, like the SmartFusion®2 and IGLOO®2 families from Microsemi, also provide additional security by encrypting the bitstream when programming is done during manufacturing. This also protects the design from being copied or reverse engineered by an unscrupulous contract manufacturer, which could also break the required Hardware Root-of-Trust.

Implementing a Secure Hardware Root-of-Trust

Once we have created a secure FPGA, the implementation of a Hardware Root-of-Trust is our next key requirement. As we saw in the requirements section, the FPGA that creates the RoT must be able to securely store and manage security keys, and implement the common cryptographic functions, usually industry standards, needed to authenticate and protect secure data transmissions. IGLOO2 and SmartFusion2 devices offer an inherently secure starting point for RoT implementations; due to their use of on-chip flash configuration storage and secure bit stream programming. Additional security features support standard encryption, decryption, and authentication operations so that communications on- and off-chip are secure as well. For a detailed description of the wide array of standard oriented security features available on IGLOO2 and SmartFusion2 devices, review the Data Security related material listed in the "To Learn More" section at the end of this paper.

As seen in the previously described secure boot process, the FPGA must protect the security keys and the phase ‘0’ immutable boot loader on-chip so that malicious intruders can’t attack or modify them in any way. When the immutable code and security keys are stored on-chip using a Flash FPGA, they also benefit from the security afforded the configuration bitstream, since they will be loaded through the configuration process. The security keys are a small portion of the overall design however, are important enough to be protected from additional forms of attack.

Protecting Security Keys from Side-Channel Attacks

As an example, one popular method of attack is to use Side-Channel Analysis (such as observations of power or timing signatures during security key related operations) to try and determine on-chip secure information. This side-channel approach is similar to one a safe cracker might use to determine the safes combination by listening to the noise made by the tumblers while manipulating the lock. In this case, the side-channel is the sound made by the physical implementation of the security “function”. SmartFusion2 and IGLOO2 FPGAs implement side-channel attack resistant decryption algorithms, and in particular are designed to be resistant to the most advanced Differential Power Analysis (DPA) form of side-channel attack.

If DPA-resistant techniques are not used, an intruder can measure the power used by the design when security keys and algorithms are being processed. Knowing the typical algorithm operations, a statistical analysis of the power use can be used to determine the security key value. For example, many security algorithms are implemented on an 8-bit basis, which means that only 256 combinations need to be checked through the DPA to identify an 8-bit section of the security key. DPA resistance can be dramatically improved by changing the architecture of the algorithm to limit such divide and conquer strategies. Additionally, changing the security key frequently will limit the number of measurements an attacker can use for statistical analysis making such approaches dramatically more difficult. Furthermore, circuit design tricks, like pre-charging registers and busses will limit the “noise” available to an intruder.

Many of the techniques used on Microsemi DPA-resistant FPGAs are licensed from Cryptographic™ Research Inc. (CRI, a division of Rambus, Inc.), and contribute to making them the most secure devices available for protecting design IP and implementing Hardware Root-of-Trust functions.

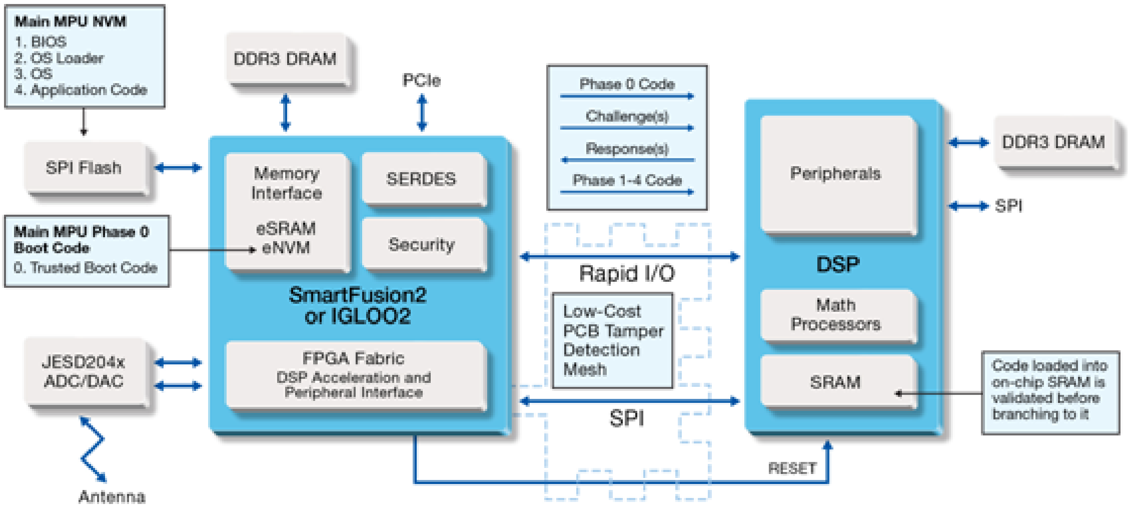


Figure 4: **Example of Secure Implementation Using SF2 or IGLOO2 FPGAs**

Implementation Description

Now that the FPGA IP has been secured and the Hardware Root-of-Trust established for our design, we can look at the full wireless communications system implementation in more detail. Figure 4 shows the implementation of a secure wireless communications system and 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 stores the balance of the DSP code (including any required OS loader and OS code along with the Application code), all of which is securely verified using a secure Challenge and Response system managed by the Root-of-Trust system. At the end of the process the secured code has been loaded into the DSP on-chip SRAM and the FPGA can allow the DSP to begin operation, confident that only authorized code is being executed. Note that a low cost PCB tamper detection mesh can be easily implemented using FPGA I/Os, so that any attempt to drill holes or cut traces can be detected and protective measures taken.

Once secure boot is completed, the FPGA can implement the other functions required by the system-bridging the PCIe and RapidIO interfaces, connecting to the JESD204x bus, pre-processing the wireless signals through the FPGA fabric, and controlling the DDR3 buffer memory. SmartFusion2 FPGA has an on-chip processor when additional algorithmic capabilities are required, while IGLOO2 FPGA can be used when the FPGA fabric is sufficient to implement the needed control functions. A critical point is that these non-security functions are already required by the system, thus the security features available in the FPGA are virtually free.

Conclusion

Flash-based FPGAs, like IGLOO2 and SmartFusion2 FPGAs, offer secure IP and Hardware Root-of-Trust that can be used to build higher-level security functions, such as secure boot, to protect your wireless communications designs from invasive intrusion. The additional security capabilities of these devices, like secured programming bitstreams, DPA resistance, and secure key storage, all come along with the fundamental architecture and design of the devices so they require no additional work or cost to the designer. With Microsemi FPGAs and SoC FPGAs advanced security is free.

References

"Thousands of Industrial Control Systems At Risk", J. Nicholas Hoover, Information Week, Jan 11, 2013

To Learn More

Data Security

1. Overview of Data Security Using Microsemi FPGAs and SoC FPGAs
2. SmartFusion2 and IGLOO2 Cryptography Services
3. Overview of Secure Boot with Microsemi SmartFusion2 SoC FPGAs

Secure Boot

1. Overview of Secure Boot with Microsemi IGLOO2 FPGAs
2. Overview of Secure Boot with Microsemi SmartFusion2 SoC FPGAs

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