Keywords: IP, piracy, FPGA, anti-piracy, theft, protect, protection, sha, sha-1,MAC, authentication, Xilinx,Altera,intellectual property

APPLICATION NOTE 4594

Protect Your FPGA Against Piracy: Cost-Effective Authentication Scheme Protects IP in SRAM-Based FPGA Designs

By: Bernhard Linke, Principal Member Technical Staff
Aug 24, 2009

Abstract: This application note describes FPGAs (field-programmable gate arrays) and how they can hold the key functions and the intellectual property (IP) of a system. It discusses ways to protect IP against piracy. SHA-1 challenge-and-response authentication is judged as the most secure methodology. This document presents a cost-effective authentication scheme that protects IP in SRAM-based FPGA designs. The DS28E01 and DS28CN01 devices are featured.

This application note appeared as an article in *Electronic Design* magazine and on the publisher's website in July, 2008.

Over the past two decades, the FPGA (field-programmable gate array) has transitioned from a prototyping tool to a flexible production solution in both consumer and industrial applications. With FPGA logic complexity increasing from a few thousand gates to millions of gates, the devices can hold more of the key functions (i.e., the intellectual property, IP) of a system.

Today, designers can select FPGAs that employ various technologies to hold the configuration data—OTP (one-time programmable) antifuses, reprogrammable flash-based storage cells, and reprogrammable SRAM-based configurable logic cells. Both antifuse- and flash-based solutions provide relatively secure solutions, since the configuration data is stored on the FPGA chip and there are mechanisms that prevent the stored data from being read out. Moreover, unless very sophisticated schemes such as depacking, microprobing, voltage contrast electron-beam microscopy, and focused-ion-beam (FIB) probing are used to pry into the silicon and to disable security mechanisms,[1] it is very unlikely that the data will be compromised. (For a short backgrounder on FPGAs, see **Appendix A**, *Technology Options and Issues for FPGAs*.)

Static-RAM-based (SRAM) FPGAs, however, have fewer safeguards to protect that IP (the configuration data) against illegal copying and theft. The reason is that once the data is loaded, it is held in SRAM memory cells, which can easily be probed to determine their contents. In addition, without some type of security mechanism to protect the configuration data before it is loaded into the chip, that data is open to snooping. Prowling through that data is possible because the bit stream is usually stored in a separate memory chip read by the FPGA at power-up when it loads its configuration pattern. There are, though, some simple ways to secure that data, ways to prevent someone from copying the configuration pattern and stealing the IP.

The Weak Point of SRAM-Based FPGAs

Due to the two-chip solution—FPGA and configuration memory—the configuration data bit stream is exposed to eavesdropping during the power-up phase. Since the FPGA cannot tell whether the bit stream is "genuine" or an illegally obtained copy, the IP contained in the configuration data is totally unprotected. This well-known issue has been partially addressed with security keys and bit-stream encryption. However, these protections are only used for high-end FPGAs, which are not suited for consumer applications due to their high cost.

Without Protection

Particularly vulnerable to design piracy are applications that use SRAM-based FPGAs without bit-stream encryption. The configuration bit stream can be captured, reprogrammed into configuration PROMs, or simply replayed to make clones of the original design. Cloned products then compete with the original product, stealing the research and development investment while reducing the market share and profitability of the original manufacturer.

Even without encryption, antifuse- or flash-based FPGAs are more secure than SRAM-based FPGAs because the configuration data is not exposed to eavesdropping. But an assembly house, if setup to program the FPGAs, can program more units than authorized and sell them on its own without incurring any development costs. Such unauthorized devices are indistinguishable from the authorized devices and can significantly impact a company's profitability.

One way to make the SRAM-based FPGAs a little more secure is to leverage multichip packaging and mount the nonvolatile memory inside a package along with the FPGA. Yet if someone opens the package, the data interface between the memory and the FPGA is exposed and the configuration pattern can be compromised.

The structure of the configuration bit stream (i.e., the sequence of data elements and how they are coded and identified) is largely undocumented. The obscurity, complexity, and size of the bit stream make the reverse-engineering process difficult and time consuming, although theoretically possible.[1] If successful, even partial reverse engineering of the configuration stream makes it possible to hack a set-top box to steal services or tamper with power-train settings in a vehicle, causing liability problems for the original manufacturer.

The Challenge...

To prevent system costs from exploding, designers must continue using SRAM-based FPGAs without encryption. However, they have to find a way to protect the IP from piracy. Furthermore, the added cost for the security measures must be as low as possible and the impact on the production flow must be minimal.

It is crucial that the security-related hardware fit into the available space on the circuit board, but not increase the overall power consumption. Also, the impact of security on the FPGA's resources (i.e., number of pins and logic elements) must be as low as possible.

And the Response: Authentication

The objective of the authentication process is to establish proof of identity between two or more entities.

Key-based authentication takes a secret key and the to-be-authenticated data (i.e., the "message") as input to compute a message authentication code (MAC). The MAC is then attached to the message. The recipient of the message performs the same computation and compares its version of the MAC to the one received with the message. If both MACs match, the message is authentic.

There is a weakness with this basic model: an intercepted message can later be replayed by a nonauthentic sender and be mistaken as authentic. The chance of a simple "replay attack" succeeding is eliminated if the MAC computation incorporates a random challenge chosen by the MAC recipient. **Figure 1** illustrates the general concept. The longer the challenge, the more difficult it is to record all possible responses for a potential replay.

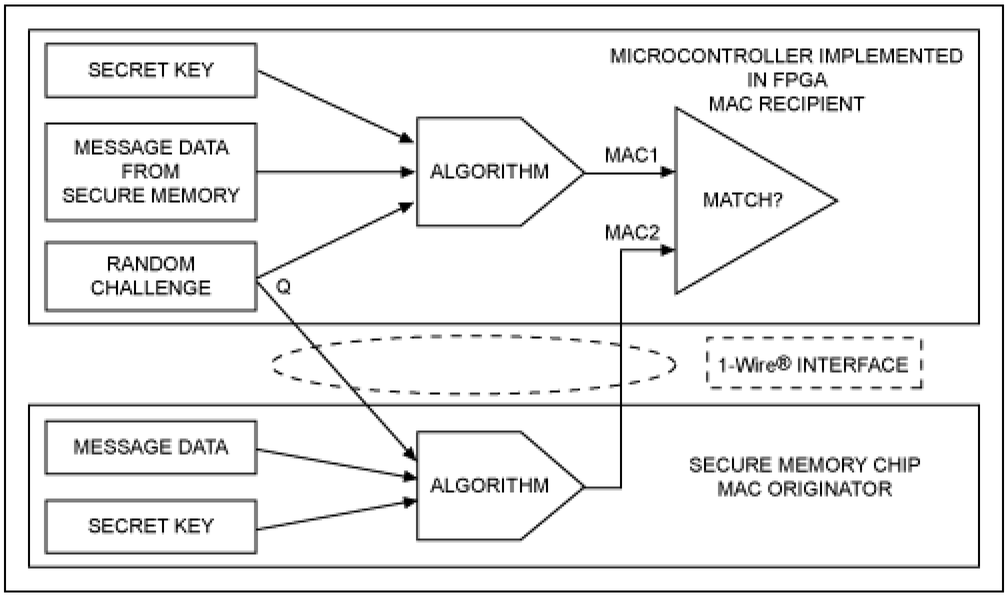
SECRET KEY
MESSAGE DATA FROM SECURE MEMORY
RANDOM CHALLENGE
Q
ALGORITHM
MAC1
MAC2
MATCH?
MICROCONTROLLER IMPLEMENTED IN FPGA MAC RECIPIENT
1-Wire® INTERFACE
MESSAGE DATA
SECRET KEY
ALGORITHM
SECURE MEMORY CHIP MAC ORIGINATOR

Figure 1. The challenge-and-response authentication process proves the authenticity of a MAC originator.

To prove the authenticity of the MAC originator, the MAC recipient generates a random number and sends it as a challenge to the originator. The MAC originator must then compute a new MAC based on the secret key, the message, and the recipient's challenge. The originator then sends the computed result back to the recipient. If the originator proves capable of generating a valid MAC for any challenge, it is very certain that it knows the secret and, therefore, can be considered authentic. The technical term for this process is challenge-and-response authentication. (See Figure 1.)

Numerous algorithms are used to compute MACs, such as Gost-Hash, HAS-160, HAVAL, MDC-2, MD2, MD4, MD5, RIPEMD, SHA family, Tiger, and WHIRLPOOL. A thoroughly scrutinized and internationally-certified one-way hash algorithm is SHA-1, which was developed by the National Institute of Standards and Technology (NIST). SHA-1 has evolved into the international standard ISO/IEC 10118-3:2004.

The math behind the SHA-1 algorithm is publicly available through the NIST website.[2] Distinctive characteristics of the SHA-1 algorithm are:

- Irreversibility: It is computationally infeasible to determine the input corresponding to a MAC.
- Collision-resistance: It is impractical to find more than one input message that produces a given MAC.
- High avalanche effect: Any change in input produces a significant change in the MAC result.

For these reasons, as well as the international scrutiny of the algorithm, SHA-1 is an excellent choice for challenge-and-response authentication of secure memories.

Hardware Implementation

A challenge-and-response authentication scheme can be implemented inexpensively as part of an SRAM-based FPGA system design (**Figure 2**). In this example, the secure memory device uses only a single pin to connect to an FPGA pin configured for bidirectional (open-drain) communication. A resistive connection to V_{DD} delivers power to the secure memory and provides the bias for open-drain communication. Maxim's DS28E01 1Kb Protected 1-Wire EEPROM with SHA-1 Engine is a good fit in this scheme. The device contains a SHA-1 engine; 128 bytes of user memory; a secret key that can be used for chip-internal operations, but cannot be read from outside; and a unique, unchangeable identification number.

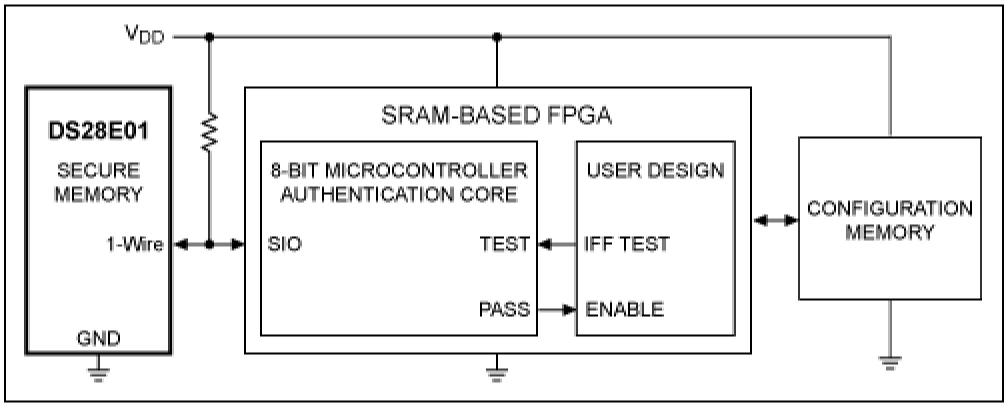

Figure 2. In this simplified schematic, a secure 1-Wire memory is used for FPGA protection.

The 1-Wire interface of the DS28E01 reduces the communications channel to just a single FPGA pin for the challenge-and-response authentication. That minimizes the impact of the security solution, since FPGAs are often I/O-pin limited. Alternate implementations can be constructed using a more generic I²C interface implemented on the FPGA and using the DS28CN01 (an I²C equivalent of the DS28E01) or by implementing the SHA-1 engine and other functions in a small ASIC or CPLD. However, using an ASIC approach would probably cost more if security is the device's only function.

To leverage the security features of the DS28E01, the FPGA must be able to do the following:

- Generate random numbers for the challenge. (On-chip random number generators usually create pseudo-random numbers, which are not as secure as real random numbers.)
- Know a secret key that can be used for internal operations, but cannot be discovered from outside.
- Compute a SHA-1 MAC that involves the secret key, a random number, and additional data, just like the secure memory.
- Compare data byte for byte, using the XOR function of the CPU implemented in the FPGA.

For detailed information on the SHA-1 MAC computation, review the Secure Hash Standard.[2] Application note 3675, "Protecting the R&D investment—two-way authentication and secure soft-feature settings" provides technical details on architecture of a secure memory and its security concept.

Microcontroller-like functionality is typically available as a free macro from major FPGA vendors. The Xilinx® microcontroller function occupies 192 logic cells, which represents just 11% of a Spartan®-3 XC3S50 device. A similar microcontroller for Altera® devices occupies 850 logic elements, which is equivalent to 18.5% of the EP2C5, the smallest member of the Cyclone® II family.

How It Works

When power is applied, the FPGA configures itself from its configuration memory. Now the FPGA's microcontroller function comes alive and performs the challenge-and-response authentication, also known as identification friend or foe (IFF). This identification involves the following steps:

- Generate a random number; send it as a challenge (Q) to the secure memory.
- Instruct the secure memory to compute a SHA-1 MAC based on its secret key, the challenge, its unique identification number, and other fixed data.
- Compute a SHA-1 MAC, i.e., the expected response (MAC1), based on the same input and constants used by the secure memory and the FPGA's secret key.
- Take the SHA-1 MAC computed by the secure memory (Read Authentication MAC) as a response (MAC2) and compare it to the expected response (MAC1).

If MAC1 and MAC2 match, the FPGA identifies its environment as a "friend," because it apparently knows the secret key. The FPGA transitions to normal operation, enabling/performing all of the functions defined in its configuration code. If the MACs are different, however, the environment must be a "foe." In this case, the FPGA takes application-specific actions rather than normal operation.

Why the Process Is Secure

Besides the inherent security provided by SHA-1, the principal security element for the above IFF authentication process is the secret key, which is not readable from the secure memory or the FPGA. Furthermore, because the data in the bit stream is scrambled, eavesdropping on the configuration bit stream when the FPGA configures itself will not reveal the secret key. Due to its size, reverse-engineering the bit stream to determine the design with the intent of removing the authentication step is very time consuming and, thus, a prohibitively difficult task.

Another critical security component is the randomness of the challenge. A predictable challenge (i.e., a constant) causes a predictable response, which can be recorded once and then replayed later by a microcontroller emulating the secure memory. With a predictable challenge, the microcontroller can effectively deceive the FPGA to consider the environment as a friend. The randomness of the challenge in this IFF approach alleviates this concern.

Security can be improved further if the secret key in each secure memory is device-specific: an individual secret key computed from a master secret, the SHA-1 memory's unique identification number, and application-specific constants. If an individual key becomes public, only a single device is affected and not the security of the entire system. To support individual secret keys, the FPGA needs to know the master secret and compute the 1-Wire SHA-1 memory chip's secret key first, before computing the expected response.

Logistical Aspects

For every unit to be built, the owner of the design (OEM) must provide one properly preprogrammed secure memory to the contract manufacturer (CM) that makes the product with the embedded FPGA. This one-to-one relationship limits the number of authorized units that the CM can build. To prevent the CM from tampering with the secure memory (e.g., claiming that additional memories are needed because some were not programmed properly), OEMs are advised to write-protect the secret key.

There is no need to worry about the security of the 1-Wire EEPROM data memory, even if it is not write protected. By design, this memory data can only be changed by individuals who know the secret key. As a welcome side effect, this characteristic lets the application designer implement soft-feature management—the FPGA can enable/disable functions depending on data that it reads from the SHA-1 secured memory.

It is not always practical for the OEM to preprogram memory devices before delivery to the CM. To address this situation, the manufacturer of the secure memory could set up a SHA-1 secret and EEPROM-array preprogramming service for the OEM. Maxim provides such a service, where secure memory devices are registered and configured at the factory according to OEM input and then shipped directly to the CM. Key benefits of such a service include:

- Eliminates the need for the OEM to disclose the secret key to the CM.
- Eliminates the need for the OEM to implement its own preprogramming system.
- Only OEM-authorized third parties have access to registered devices.
- The vendor maintains records of shipped quantities, if needed for OEM auditing.

Proof of Concept

The FPGA security method featured in this application note has been tested with products from Altera and Xilinx. In its white paper, "An FPGA Design Security Solution Using a Secure Memory Device," Altera concludes: "This FPGA design security IFF solution protects Altera FPGA designs from being cloned even if the configuration data bit stream is captured. The user design remains disabled until the hash algorithm computation in both the FPGA and the secure memory match. This design security solution protects FPGA designer's IP."[3]

Similarly, Xilinx states in its application note XAPP780: "The system's security is fundamentally based on the secrecy of the secret key and loading of the key in a secure environment. This entire reference design, except the secret key, is public abiding by the widely accepted Kerckhoffs' law. The simple interface to programming and authentication provided in this application note make this copy protection scheme very easy to implement."[4] (In his groundbreaking article on military cryptography, the Flemish linguist Auguste Kerckhoffs argued that instead of relying on obscurity, security should depend on the strength of keys. He argued that in the event of a breach, only the keys would need to be replaced instead of the whole system.)

Conclusion

Protection against piracy of IP requires adding just one low-cost chip like the DS28E01 and updates to the FPGA configuration code. Thanks to the 1-Wire interface, only a single FPGA pin is used for security purposes. If more pins are available on the FPGA, the I^{2}C version of the secure memory can be used in lieu of the 1-Wire version. This would also necessitate some changes to the FPGA configuration pattern as well as the control software for the embedded microcontroller.

The secure memories can be ordered preprogrammed with either a fixed or computed secret key and application-specific data. Preprogrammed parts then get shipped only to the OEM, or possibly to authorized CMs. The CMs can only build as many units as preprogrammed parts are supplied.

Appendix A

Technology Options and Issues for FPGAs

The five mainstream FPGA vendors—Actel®, Altera, Lattice Semiconductor Corporation®, QuickLogic®, and Xilinx—have a combined market share of approximately 98%. The remaining percentage points account for a few specialty suppliers that offer FPGA-like capabilities.

All of these companies are fabless operations and rely on wafer foundries in Taiwan, Japan, Singapore, or Germany to produce their wafers. As a consequence, they only have access to the technologies of their foundries, such as fast logic with SRAM capability or high-density floating-gate memory arrays. By leveraging standard process flows, FPGA companies can keep their manufacturing costs low.

Actel and QuickLogic, however, worked with their foundries to include proprietary antifuse technologies in the process flow. The antifuse technology provides some unique benefits: good security, small cell size, radiation tolerance, and, of course, nonvolatility. Still, antifuse technology has one limitation. Unlike flash-based devices, once the cells are configured, they cannot be reconfigured (as is indicated by "fuse"). In addition, although the antifuse process is more complex than SRAM-based FPGAs, the cells are smaller, which results in smaller chips and lower cost for equivalent logic capacities.

For a long time, SRAM and flash technologies were mutually exclusive; they could not easily be integrated into one chip. Yet this scenario has changed, thanks to recent developments in semiconductor technology driven by market demand. The smaller competitors (Actel, Lattice Semiconductor, and QuickLogic) now offer flash-based, single-chip FPGAs (the ProASIC, Lattice®XP2, and PolarPro families, respectively).

Altera, for example, has offered flash-based CPLDs (MAXII) since 2004, but currently does not use this technology for FPGAs. Xilinx introduced its Spartan-3AN product line of flash-based FPGAs early in 2007, mounting two chips (FPGA and flash memory) in a single package. Xilinx does not, however, have a monolithic flash-based FPGA product.[5] The ability to co-integrate the flash storage provides an improvement in IP security over the SRAM-based FPGAs, which use an external configuration memory. With co-integration on a single chip, it is harder for someone to snoop the data transfer between the memory and configurable logic array.

One approach that meets all of these security requirements involves a challenge-and-response data exchange (authentication) between the FPGA and a secure memory. Secure-memory chips are an invention of the late 1980s. They became popular in Europe initially for payphone cards, and in the 1990s for bank cards; they are now a vital component in GSM cell phones (i.e., the SIM card). The prevalent communication standard between the secure chip cards and a host system is the I²C serial bus.

Custom-tailored for either banking or phone applications, secure memories were not considered viable for general-purpose use. This changed back in 2000 when Dallas Semiconductor (now Maxim Integrated Products) introduced devices that incorporate the SHA-1 hash algorithm. The first-generation DS2432 was followed by the enhanced DS28E01. These devices use the 1-Wire interface, which serves for both communication and power supply. The DS28CN01, introduced in 2007, uses an I²C interface, but is otherwise similar to the DS28E01.

Another aspect to consider regarding IP security is the integrity of the foundries, since the foundries often have intimate knowledge of the FPGA design. Trust or strict control and supervision are necessary to protect proprietary information from getting into the wrong hands. This might be easier to achieve with domestic foundries than those abroad. Thus far, though, the major foundries have demonstrated excellent integrity in securing design details.

References

1. Drimer, Saar, "Volatile FPGA design security — a survey," work in progress, http://www.cl.cam.ac.uk/~sd410/papers/fpga_security.pdf.
2. Secure Hash Standard, http://www.itl.nist.gov/fipspubs/fip180-1.htm.
3. Altera White Paper 01033, "An FPGA Design Security Solution Using a Secure Memory Device,"

http://www.altera.com/literature/wp/wp-01033.pdf.

4. Xilinx Application Note XAPP780, "FPGA IFF Copy Protection Using Dallas Semiconductor/Maxim DS2432 Secure EEPROMs," http://www.xilinx.com/support/documentation/application_notes/xapp780.pdf.
5. FPGA and Structured ASIC Journal, February 27, 2007, "Short Stack with Syrup," https://data.kemt.fei.tuke.sk/ProgramovatelneLogickeObvody/_materialy/prednasky/SECURITY/20070227_stack.pdf.

Related Parts		
DS28CN01	1Kbit I²C/SMBus EEPROM with SHA-1 Engine	Free Samples
DS28E01-100	1Kb Protected 1-Wire EEPROM with SHA-1 Engine	Free Samples
DS28E02	1-Wire SHA-1 Authenticated 1Kb EEPROM with 1.8V Operation	Free Samples
DS28E10	1-Wire SHA-1 Authenticator	Free Samples