XAPP780 (v1.1) May 28, 2010

FPGA IFF Copy Protection Using Dallas Semiconductor/Maxim DS2432 Secure EEPROMs

Author: Catalin Baetoniu

Summary

This application note describes a cost-optimized copy protection scheme that helps protect an FPGA against cloning. The design leverages an external secure serial EEPROM, such as the Dallas Semiconductor/Maxim DS2432 (1 kbit protected 1-wire EEPROM with SHA-1 engine). The included reference design uses an optimized PicoBlaze™ 8-bit microcontroller. This application note provides a hardware design with associated PicoBlaze software code. The code loads a secret key into the secure EEPROM and authenticates the user system with the secure EEPROM.

Note: This application note is provided for demonstration purposes only. Its design might be vulnerable to security attacks on the configuration bitstream. Contact your local Xilinx sales office or Xilinx distributor for an updated design. Contact xapp780@xilinx.com for questions and feedback on this application note.

Introduction

Many FPGA designs require copy protection to protect FPGA intellectual property from cloning by unauthorized parties. There are many levels of security that can be applied in terms of copy protection.

Virtex®-II, Virtex-II Pro, Virtex-II Pro X, Virtex-4, Virtex-5, Virtex-6, and a few of the larger Spartan®-6 devices offer *encryption* of the configuration data which is used in systems requiring the highest level of security. For more cost-sensitive applications where the level of security required is limited to protecting against unauthorized cloning of FPGA devices, *authentication* using an *Identification Friend or Foe (IFF)* concept is viable. This concept can be applied to all FPGA families, including the low-cost Spartan-3 Generation series. Reference design files are provided for Spartan-3, Spartan-3A, Spartan-6, Virtex-II Pro, and Virtex-5 FPGAs.

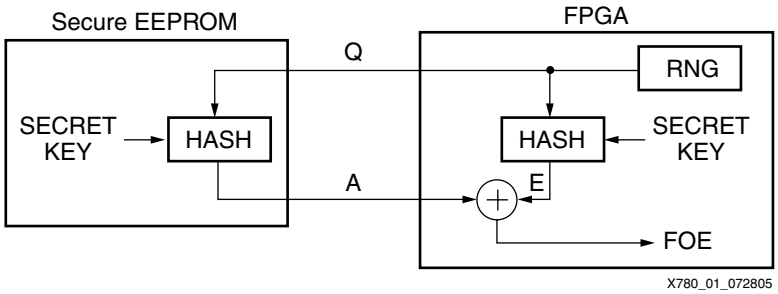

Figure 1: **Identification Friend or Foe (IFF)**

Figure 1 shows the IFF concept. The following steps determine if the system has been identified as a Friend or Foe:

1. First, the FPGA implements a Random Number Generator (RNG) that produces a random message **Q,** which is sent to the secure EEPROM.
2. The secure EEPROM uses a secret key, known only to the designer, and a hash function to encrypt the message **Q** and produce the response **A**.
3. The FPGA uses the same secret key to determine the expected response **E,** and compares it with the actual responses **A**, coming from the secure EEPROM.

4. If the expected and actual responses match, the design is a Friend. Otherwise the design is labeled as a Foe because tampering might have occurred with the system.
5. Finally, the FPGA application must be designed such that if a Foe is detected, the application ceases to operate or operates with diminished functionality, such as a demo mode. The user design functions correctly only when the system is detected as a Friend.

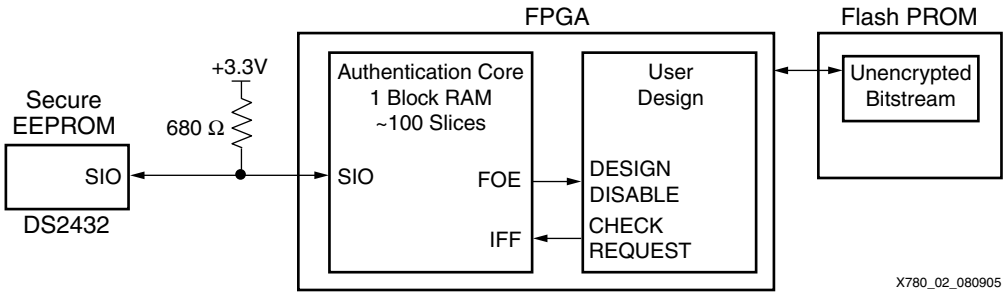

Figure 2: **DS2432 Secure EEPROM Connectivity to Xilinx FPGA**

This application note uses a Dallas Semiconductor/Maxim DS2432 secure EEPROM. However, the IFF concept can also be applied to other secure EEPROMs. Figure 2 shows the connectivity diagram between the DS2432 secure EEPROM and a Xilinx FPGA to implement the copy protection scheme outlined in this application note. First, the FPGA configures itself from a flash PROM. When the FPGA is configured, the user design is automatically disabled until it authenticates with the secure EEPROM using a secret key that is stored in the FPGA against the stored encrypted key in the secure EEPROM.

The features of the DS2432 secure EEPROM include:

- 64-bit read-only unique serial number (no two devices share the same ID)
- 64-bit write-only secret key that can be rewritten at any time, but there is no way of reading it back
- Secure Hash Algorithm (SHA-1) cryptographic engine
- Serial 1 wire interface for low pin count

The widely accepted security principle called *Kerckhoffs' law* states that security cannot be achieved through obscurity, and a cryptosystem is considered secure if a cryptographically secure algorithm is used and the details of the design are public. According to the principle, the security of the design is ensured by keeping a master key secret. The 64-bit key should be kept in a secure environment within the design center and should not be released to third parties, such as contract manufacturers.

Hardware System

The reference design provides two designs, LOADTEST and IFFTEST, that assist in designing the DS2432 into a system for copy protection. IFFTEST is used after a valid authentication key has been programmed using LOADTEST. Table 1 indicates how each design is used in the prototyping, manufacturing, and production environment.

Table 1: **XAPP780 Reference Systems Usage**

	LOADTEST	IFFTEST
Prototyping	Used to load the authentication key (hash of the secret key and a unique serial number) into the DS2432 chip.	Example of a copy protection design to authenticate a system with a valid DS2432 device. This outputs an error and should disable the user design if the secure EEPROM is not loaded with the correct authentication key.
Manufacturing	Only used in a secure environment to program the key.	Not used in the manufacturing environment.
Production	Do not include in the final user design.	Embedded in the final user design to authenticate the DS2432 device that has been loaded with an authentication key.

The hardware systems are built on a modified PicoBlaze module (KCPSM2) described in Xilinx application note XAPP627. Only two modifications need to be made to this application note's design: the register file must be modified and a second port is added on the block RAM for 128 bytes of scratchpad memory.

LOADTEST

The loading of the secret key must be done in a secure environment, and the secret key MUST be protected from information leaks. Loading the key into the DS2432 device is done through the LOADTEST design, shown in Figure 3. This LOADTEST design is NOT to be included in the final user design but simply used in the secure environment. In this design, when the PRGM pin sees an active-High edge, the loading core attempts to program the key into the DS2432 device. If the key is stored successfully, the PASS signal goes High. Otherwise the PASS signal remains Low.

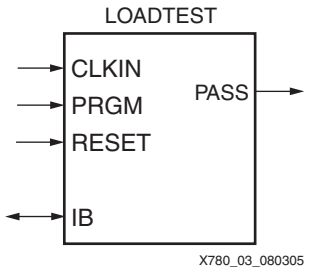

Figure 3: **LOADTEST Instantiation Block Diagram**

The inputs to the LOADTEST design are:

- An input clock named CLKIN, with a minimum frequency of 20 MHz. The actual frequency of this clock must be set within the LOADTEST HDL file as shown in Figure 4. The clock period must be adjusted in the UCF file to ensure that timing is met.
- An active-High RESET signal is provided for the authentication core.
- The active-High PRGM signal activates the loading of the secret key in the DS2432 device.

VHDL

```
-- MAXCNT must be equal to your system clock frequency expressed in MHz
constant MAXCNT:INTEGER:=50; -- for a 50 MHz CLK
```

Figure 4: **Required Changes in LOADTEST or IFFTEST to Specify the Correct Clock Frequency**

The output from the design is the PASS signal, which is Low by default. When the secret key is successfully loaded into the DS2432 device, the PASS signal goes High. The IB signal is the bidirectional DS2432 interface pin to the design.

Fundamental to the authentication scheme is the need to use a secret key. Replace the secret key used in the reference design with your own secret key. The 0x0123_4567_89AB_CDEF value is the first one an attacker will use to try to break the copy protection.

In this IFF authentication application, the secret key is not directly stored in the DS2432 device. LOADTEST stores a hash of the secret key and the device's unique serial number in the DS2432 device. Therefore, the authentication key stored in each DS2432 device is different even though the secret key is the same, providing increased security because every secure EEPROM appears to be different.

Figure 5 shows the HDL lines that need to be changed in the LOADTEST project to change the secret key.

VHDL

```
-- This is the secret master key - replace it with your own!
constant KEY:TKEY:=(X"01",X"23",X"45",X"67",X"89",X"AB",X"CD",X"EF");
```

Figure 5: **Required Changes in LOADTEST or IFFTEST to Specify a Secret Key**

After the secret key is loaded successfully, the system is ready to be authenticated using the IFFTEST design.

IFFTEST

Use the IFFTEST design to authenticate a system, shown in Figure 6, as part of the final user design. The basic concept is that a user design is deactivated by default, and then it tries to authenticate the system. If a valid secure EEPROM is found with the correct secret key, the user design is activated.

To authenticate the system, the user design asserts the IFF signal on the IFFTEST core from Low to High. If the system integrity has been validated, the Foe signal goes Low indicating the secure EEPROM is a Friend and not a Foe. If the system integrity is altered, the Foe signal remains High.

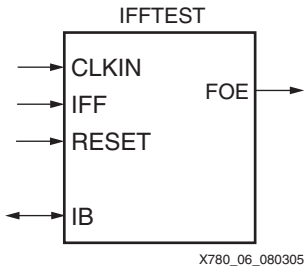

Figure 6: **IFFTEST Instantiation Block Diagram**

The inputs to the IFFTEST design are:

- An input clock named CLKIN, with a minimum frequency of 20 MHz. The actual frequency of this clock must be set within the LOADTEST HDL file as shown in Figure 4. The clock period must be adjusted in the UCF file to ensure that timing is met.
- An active-High RESET signal is provided for the authentication core.
- A rising edge on the IFF signal initiates the authentication of the DS2432 device.

The FOE output signal disables the user design when High and enables the user design when Low. The IB signal is the bidirectional DS2432 interface pin to the design.

Figure 7 shows the HDL instantiations for the IFFTEST core within the user design.

VHDL

```
component IFFTEST
   port(IFF:in STD_LOGIC;
       FOE:out STD_LOGIC;
       CLKIN:in STD_LOGIC;
       RESET:in STD_LOGIC;
       IB:inout STD_LOGIC);
end component;

IFF_inst: IFFTEST
   port map(IFF => IFF_signal,
       FOE => FOE_signal,
       CLKIN => CLKIN_signal,
       RESET => RESET_signal,
       IB => IB_signal);
```

Figure 7: **HDL Instantiation of IFFTEST**

To authenticate the system, the secret key specified in IFFTEST must match the secret key in LOADTEST. Replace the secret key with your own secret key as shown in Figure 5, page 4.

Software

To use the design as-is, no software changes are required for this design.

For PicoBlaze users who wish to modify the software content, the commented PicoBlaze assembly code is provided in `loadtest.psm` and `ifftest.psm`. To compile the code, the updated ROM_form files must be included in the directory where the KCPSM2 compiler is run. The updated ROM_form files are necessary to allow the PicoBlaze core to use the second memory port as scratchpad memory. Refer to the “PicoBlaze Assembler” section of XAPP627 for more details on how to use the PicoBlaze compiler.

Conclusion

This application note describes a low-cost method to protect the FPGA from cloning with a secure EEPROM using the concept of authentication. 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 into a system.

Design Resources

The reference design described in this application note can be downloaded from the following link:

https://secure.xilinx.com/webreg/clickthrough.do?cid=55693

References

The following documents provide supplementary material useful with this application note:

1. Xilinx application note XAPP627: “PicoBlaze 8-Bit Microcontroller for Virtex-II Series Devices”
   http://www.xilinx.com/support/documentation/application_notes/xapp627.pdf
2. Kerckhoffs' law
   http://en.wikipedia.org/wiki/Kerckhoffs_law
3. Dallas Semiconductor/Maxim DS2432 data sheet
   http://pdfserv.maxim-ic.com/en/ds/DS2432.pdf

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
08/17/05	1.0	Initial Xilinx release.
05/28/10	1.1	Removed Shalin Sheth as document author. In “Introduction,” updated list of devices offering encryption and added a sentence to specify FPGA devices supported by reference designs.

Notice of Disclaimer