Appears in *Proceedings: 15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines – FCCM 2007,*
IEEE, pp. 287–288, ISBN-10 0-7695-2940-2, ISBN-13 978-0-7695-2940-0.

New Protection Mechanisms for Intellectual Property in Reconfigurable Logic

Tim Güneysu, Bodo Möller, Christof Paar
Horst Görtz Institute for IT Security, Ruhr University Bochum, Germany
{gueneysu,bmoeller,cpaar}@crypto.rub.de

Abstract

The distinct advantage of SRAM-based Field Programmable Gate Arrays (FPGA) is their flexibility for configuration changes. But this opens up the threat of Intellectual Property (IP) theft since the system configuration is stored in easy-to-access Flash memory. High-end FPGAs have already been extended with symmetric-key decryption engines used to load an encrypted version of the configuration that cannot simply be copied and used without knowledge of the secret key. However, with respect to business and licensing processes, this protection system lacks a convenient scheme for key transport and installation.

We propose a new protection scheme for the IP of circuits in configuration bit files that provides a significant improvement to the current unsatisfying situation. It uses both public-key and symmetric cryptography, but does not burden FPGAs with the usual overhead of public-key cryptography: While it needs hard-wired symmetric cryptography, the public-key functionality is moved into a temporary configuration bit stream for a one-time setup procedure. This approach requires only very few modifications to current FPGA technology. Using five basic stages, the new protection scheme allows new accounting models for volume licensing of IP, with automated key installation on FPGAs taking place at the customer's site.

IP Protection for FPGAs

When Field Programmable Gate Arrays (FPGA) were first introduced in the 1980s, this was a revolutionary step from static ASIC and VLSI solutions to flexible and maintainable hardware applications. It has become possible to avoid the static designs of standard VLSI technology, and instead to compile electrical circuits for arbitrary hardware functions into configuration bit files used to program a fabric of reconfigurable logic. However, the flexibility of SRAM-based FPGAs also brings up the issue of protecting the Intellectual Property (IP) of such circuit layouts from unauthorized duplication or reverse engineering. Unfortunately, a configuration bit file of an FPGA can easily be retrieved from a product and used to clone a device with only little effort.

To cope with these problems, various approaches have been proposed – for example, based on the exchange of cryptographic handshake tokens [1]. On high-end devices, FPGA manufacturers such as Xilinx and Altera now allow the use of encrypted bit streams. Unfortunately, these approaches do not offer a full protection scheme to prevent IP cloning. A complete protection system based on bit stream encryption has been proposed by Kean [3]. Kean's proposal requires the implementation of additional security features in the FPGA. It also requires the participation of the FPGA manufacturer (or a trusted party) whenever a bit stream is to be encrypted for a particular FPGA, meaning that such transactions can't be kept just between the IP vendor and the customer.

Our Contribution. We propose a new protection scheme for configuration bit files. As in Kean's proposal, bit streams are encrypted for individual FPGAs, allowing the IP vendor to exactly track the potential use of their licensed designs. Unlike Kean's proposal, our approach does not require the continuing participation of a third party, allowing for off-site IP installation without any interaction with the FPGA manufacturer. To enable FPGAs for these new features, only very few modification are required compared with recently available FPGA models.

Our idea assumes three participating business parties. The first contributor is the trusted Hardware Manufacturer (HM) who designs and creates FPGA devices. A second participant is the Intellectual Property Owner (IPO) who has created some novel logic design for a specific problem. This IP is synthesized as a configuration bit file for a specific class of FGPAs manufactured and provided by the HM. The IPO wants to distribute the configuration bit file using a special cost or licensing model, usually on a per-volume basis. The final participant is a System Integrator (SI) who intends to use the IPO's design in products employing the HM's FPGA devices. For example, based on a volume licensing model, the SI must pay a license fee for each product that includes the design.

In our scheme, the FPGA manufacturer (i.e., HM) creates a personalization module (PM) when deploying a new class of FPGAs on the market. The PM is a specific bit stream configuring the FPGA fabric with functionality for public-key cryptography. The Diffie-Hellman scheme over elliptic curves (ECDH) with a Key Derivation Function (KDF) can be used to establish keys for symmetric cryptography (see [4] for scheme details). The PM contains a static ECDH secret key created by the HM (common to the FPGA class) and provides an interface to receive an ECDH public key. This public key is assumed to come from the IPO, thus allowing the PM to internally compute an ECDH result that can also be computed by the IPO. The symmetric key material can easily be bound to an individual FPGA by using a static identification assigned to the FPGA during manufacture; the PM has to provide some interface (e.g., JTAG) to extract this device ID so that it can be sent to the IPO. The PM uses a dedicated interface (e.g., SelectMap) to write the resulting symmetric key material to the key storage of the bit stream decryption engine (a feature already available on many FPGAs).

Thus, we can set up each FPGA with an individual symmetric key depending on the IPO's public key and on the FPGA's device ID. The overhead in terms of additional security components on the FPGA is negligible. The ECDH functionality is no permanent burden as it is needed just once for personalization and uses the fabric of the FPGA. Our scheme uses five main stages:

A. **SETUP.** The protection setup is performed once by the HM when deploying a new class of FPGAs. The HM creates a personalization module as described above, publishing the ECDH public key P_{HM} corresponding to the PM's ECDH secret key. Then, the HM symmetrically encrypts this PM using a key included in every FPGA during manufacture, publishing the encrypted result P_{enc}.

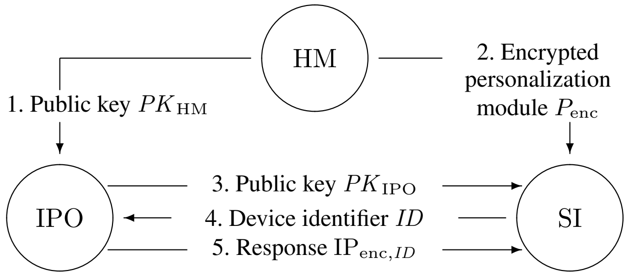
B. **LICENSING.** When an IPO offer its IP to an SI, it provides a public key PK_{IPO} of its own.

C. **PERSONALIZATION.** The SI uses the personalization module (available via P_{enc}) on each FPGA it intends to use, providing PK_{IPO} to the FPGA and extracting the respective device ID. At this stage, the FPGA computes symmetric key material. Henceforth, it can use this key material in its internal bit-stream decryption engine.

D. **CONFIGURATION.** The SI sends the device ID to the IPO. To issue a license, the IPO generates a specific configuration file $\mathrm{IP}_{\mathrm{enc},ID}$ for the individual FPGA (using the public key P_{HM} and its own ECDH secret key to derive the same key material that the FPGA obtained during personalization).

E. **INSTALLATION.** The SI installs $\mathrm{IP}_{\mathrm{enc},ID}$ in the FPGA.

The message flow between the parties is as follows:


HM
1. Public key PK_{HM}
2. Encrypted personalization module P_{enc}
IPO
3. Public key PK_{IPO}
4. Device identifier ID
5. Response $\mathrm{IP}_{\mathrm{enc},ID}$
SI

A personalization module as sketched above is highly feasible: given a public-key cryptography core by Altera [2] with only 300 logical elements (LE, assumed to consist of a four-input look-up-table and a flip-flop), the personalization module including overhead can be estimated with a logical area size of less than 1000 LEs of the FPGA. This is small enough to fit all recent FPGA types (e.g., the smallest Xilinx Virtex 4 XC4VFX12 already provides 12,312 LEs) and leaves enough free space to add functionality to provide tamper resistance and fault tolerance. If sufficient fabric area remains to implement a physical random number generator in the PM, a variant of our scheme without a permanent device ID can be used.

References

[1] Altera Corporation. FPGA design security using MAX II reference design. URL `http://www.altera.com/end-markets/refdesigns/sys-sol/indust_mil/ref-des-secur.html`.

[2] J. Fry and M. Langhammer. RSA & Public Key Cryptography in FPGAs. Technical report, Altera Corporation, 2005.

[3] T. Kean. Cryptographic Rights Management of FPGA Intellectual Property Cores. In *Proceedings ACM Conference on FPGAs*, Monterey, CA, 2002.

[4] National Institute of Standards and Technology (NIST). Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography. NIST Special Publication SP 800-56A, 2006.