A Pay-per-Use Licensing Scheme for Hardware IP Cores in Recent SRAM-Based FPGAs

Roel Maes, *Student Member, IEEE*, Dries Schellekens, and Ingrid Verbauwhede, *Senior Member, IEEE*

***Abstract*—Currently achievable intellectual property (IP) protection solutions for field-programmable gate arrays (FPGAs) are limited to single large "monolithic" configurations. However, the ever growing capabilities of FPGAs and the consequential increasing complexity of their designs ask for a modular development model, where individual IP cores from multiple parties are integrated into a larger system. To enable such a model, the availability of IP protection at the modular level is imperative. In this work, we propose an IP protection mechanism for FPGA designs at the level of individual IP cores, by making use of the self-reconfiguring capabilities of modern FPGAs and deploying a trusted third party to run a metering service, similar to the work of Güneysu *et al.* and Drimer *et al.* The proposed scheme makes it possible to enforce a pay-per-use licensing scheme which holds considerable advantages, both for IP core providers as well as for system integrators. Moreover, the scheme has a minimal implementation overhead and is the first of its kind to be solely based on primitives that are already available in recent commercially available FPGA devices. This allows for an immediate and feasible deployment, in contrast to earlier proposed solutions.**

***Index Terms*—Cloning, design security, field-programmable gate array (FPGA), hardware metering, intellectual property (IP) protection, reverse-engineering, soft intellectual property (IP).**

I. Introduction

FIELD-PROGRAMMABLE gate arrays (FPGAs) are the principal type of reconfigurable logic integrated circuits. The key idea of providing "in-the-field" (re)configurable hardware primitives offers considerable advantages for design cost and flexibility, time-to-market, unit price, etc., compared to traditional ASICs. This is why FPGAs are considered a game-changer in the silicon industry and their use is still rapidly gaining in popularity; e.g., since recently, Intel is selling a version of its Atom processor paired with an FPGA to increase flexibility [3]. In recent years, manufacturing progress has also enabled ever bigger and faster devices, with the very latest high-end FPGAs containing well over one million logic cells and an aggregate I/O bandwidth well over 1 Tb/s [4], [5]. On the other hand, designing for such powerful devices becomes an incredibly complex task. Earlier reconfigurable devices implemented relatively small single-task circuits which were often designed by a single party using only one or a few functional blocks and simple interfaces. However, modern FPGAs can support entire high-end digital systems incorporating a multitude of modules and requiring advanced I/O interfaces. Developing such systems from scratch has become an insurmountable task for most developers and a system-level design approach (re)using custom and third-party intellectual property (IP) modules has become standard procedure. FPGA vendors recognize this evolution and in recent versions of their development tools such a modular design approach is more and more supported, e.g., the Xilinx Plug-and-Play IP Initiative [6]. This is an important step towards enabling an FPGA IP core market, wherein IP developers can build a business case selling separate IP blocks and system developers can buy IP modules from different vendors to integrate them in a system design.

Another critical issue which surfaced with the rise of increasingly more complex and hence more valuable FPGA designs is the protection of the IP contained in them. The very flexible and volatile nature of an FPGA configuration is the key to many of the advantages associated with FPGAs, but also opens the door to IP abuse; e.g., copying an FPGA design is easy since an FPGA's configuration is effectively digital data. The FPGA vendors' main answer to this is to support configuration encryption by providing an on-chip cryptographic decryption module and secure key storage. This method is very effective against direct cloning of FPGA configurations in commercial end products, but its use is rather rigid and sacrifices part of the FPGAs flexibility. The lack of a key management service makes installing the decryption keys in the devices a tedious and security-sensitive task for a system developer. Moreover, the offered protection is not adapted to the modular design approach discussed earlier, since only a single "monolithic" FPGA design can be encrypted. In a system-level FPGA design environment, integrating many IP cores from different parties, the need for IP protection at a modular level, in addition to the system level, is evident.

A. Our Contribution

In the work at hand, we make use of the existing protection primitives available on commercial FPGAs to build a more elaborate and feasible construction which offers IP protection both at the module and at the system level. The main concepts which enable the proposed scheme are the use of a trusted third party to provide an independent metering service between IP providers, system developers and customers, and the use of the self-reconfiguring techniques of modern FPGA devices to enhance the FPGA's protection primitives. Similar protocols based on these

Manuscript received January 20, 2011; revised July 30, 2011; accepted September 06, 2011. Date of publication September 29, 2011; date of current version January 13, 2012. COSIC is a member of IBBT. This work was supported in part by IAP Program P6/26 BCRYPT of the Belgian State and by K.U.Leuven-GOA funding. The work of R. Maes was supported by an IWT-Vlaanderen Ph.D. fellowship (71369). The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ramesh Karri.

The authors are with the COSIC Research Group, Department of Electrical Engineering (ESAT), Katholieke Universiteit Leuven, 3001 Heverlee, Belgium (e-mail: roel.maes@esat.kuleuven.be; dries.schellekens@esat.kuleuven.be; ingrid.verbauwhede@esat.kuleuven.be).

Digital Object Identifier 10.1109/TIFS.2011.2169667

techniques have been proposed earlier [1], [2], but we specifically aim for a scheme which is more practical and can be deployed on currently existing devices. Moreover, our proposal allows us to enforce a "pay-per-use" licensing scheme where system developers pay a small amount to the IP provider in order to use a particular module only once, instead of a large sum to use it indefinitely. This holds advantages, both for the system developer and the IP owner. The IP owner keeps full control over the use of its IP cores and is protected from unlicensed over-use or redistribution, whereas system developers who could not afford the expensive unlimited IP license are now able to obtain a number of single instances of the required IP cores at a much lower cost.

B. Outline

This work begins with a detailed introduction into the FPGA design security problem in Section II and an overview of earlier proposed solutions in Section III. The main contribution of this work, a pay-per-instance active metering scheme for FPGA designs, is provided in Section IV. Finally, some aspects of the proposed scheme are discussed in more detail in Section V.

II. FPGA Design Security

A. FPGA Basics

An FPGA consists of a large array of configurable logic primitives, such as lookup tables, registers and memory blocks, and a configuration memory for storing their configuration. FPGA types can be distinguished based on the nature of their configuration memory. Nonvolatile FPGAs are relatively costly and are intended for applications where reliability is critical, e.g., in military and aerospace. Most commercial FPGA applications make use of volatile SRAM-based FPGAs which are the focus of this work.

An SRAM-based FPGA contains a dedicated controller which loads design data into the configuration memory. The design data is applied in a binary format called the *bit stream*, which can be interpreted by this configuration controller. The FPGA vendor's design tools translate high-level design descriptions (HDL) into digital bit streams. The bit stream's format is documented (e.g., [7]), but the exact interpretation of the configuration data is not public and proprietary to the FPGA vendor. Although there are no known methods to completely reverse-engineer bit streams, extracting useful information about the design from the bit stream data is not difficult [8]. Relying on the *obfuscation* offered by the bit stream's encoding to protect valuable or secret design information is considered risky and will be avoided in this work.

B. FPGA Design Security Issues

Due to their digital nature, obtaining, investigating, and in particular duplicating an FPGA's bit stream is much easier for an adversary than attacking a hard-wired ASIC design. Based on his goals, a number of different attacker scenarios targeting electronic hardware products in general and FPGA products in particular can be distinguished.

Cloning: occurs when an adversary creates an exact copy or *clone* of the original product. The cloned product can be sold under a different label, or even under the same label as the genuine producer. The latter is generally called *counterfeiting*. Since the cloner bears minimal engineering costs and time-to-market, he has a large and unfair market advantage. The risks of cloning to the genuine manufacturer include reduced profits and market share, but also brand damage due to the decreased reliability of the cloned products, leading to early product failures and safety hazards. Since unconfigured FPGA devices are off-the-shelve products and digital bit streams are easy to eavesdrop and duplicate, FPGA designs are particularly sensitive to cloning. It is clear that cloning is prohibited by law, as stated by many national and international IP protection laws and directives.

Overbuilding: is a special type of cloning, closely related to an outsourced production model. Overbuilding is unauthorized overproduction of the outsourced product, by the outsourcee or by an illicit sister company. The overbuilt products are sold through alternative channels (black market), usually at a lower cost. Overbuilding holds the same risks as regular cloning. Preventing or reacting to overbuilding (or cloning in general) on a legal basis can be difficult, in particular when it happens offshore.

Reverse Engineering: is the act of analyzing an existing product, be it an end product or a (soft) IP block, in order to learn and reuse any innovative elements such as algorithm optimizations, design decisions, and implementation strategies. Doing this, an adversary can (re)create a competing product at a much smaller research and development cost, which offers him an unfair market advantage. Another goal of reverse engineering is to bypass existing security measures, e.g., put in place to protect the IP in the first place. The original designer suffers from reduced income due to reverse engineering, but possibly also from disclosure of his IP. Relying on the bit stream's obfuscation to protect against reverse engineering is risky and does not offer any level of cryptographic security.

Tampering: is an extension of reverse engineering, where the adversary also makes modifications to the design, e.g., to gain unauthorized access to the product, to steal secret or protected data stored or communicated by the device, or to sabotage the functionality. A notable example of tampering is the addition of *hardware Trojans* [9]. Tampering is of particular concern to parties relying on the integrity of their product, e.g., military, banks, pay-TV providers, etc.

Note that very similar problems are encountered in software security. For software, protection measures are implemented at a lower functional level, i.e., in the operating system or in the code-executing hardware, e.g., the Trusted Computing initiative [10]. For FPGA designs, such measures are not possible since there are no underlying operation levels besides the actual silicon. Any FPGA bit stream protection should originate from the supporting hardware.

Adversaries can also be differentiated based on their capabilities, i.e., the knowledge, tools, funds, etc. they are able to invest in order to achieve their goal. No feasible IP protection measure offers absolute security but should be evaluated with

respect to the minimal required effort in order to bypass it. A protection measure is adequate if the assumed effort to break it supersedes the expected benefits of the attack scenario. The classic taxonomy [11] for hardware security distinguishes between low class, middle class, and high class attacker profiles. In reality, a more fine-grained classification might be necessary.

A Low Class Adversary (Clever Outsider): has only logical access to the FPGA and possibly the development tools, documentation, and generated files. He has no insider knowledge of the used systems and cannot afford methods to gain physical access to the internals of an integrated circuit. Assumably, a low class adversary is able to extract design information from a plaintext bit stream [8].[1]

A Middle Class Adversary (Knowledgeable Insider): has varying degrees of knowledge about and access to the internals of the product. A typical example is the level of access a system developer has to an external soft IP block that he purchased, depending on the format of the IP block, or the insider knowledge an FPGA vendor has about the exact details of its proprietary bit stream format. Probing internal signals on modern submicrometer FPGAs is beyond his capabilities.

A High Class Adversary (Funded Organization): has top-level expertise in all the required fields and disposes of the most high-end tools for attacking the product. He can also gain physical access to the internals of an FPGA such as on-chip busses and memories [12].

The goal of this work is to realize a feasible modular soft IP core protection scheme which prevents low and middle class adversaries of cloning (and hence overbuilding) and reverse-engineering obtained cores.

C. Parties Involved

Part of the complexity of the FPGA design security problem are the different entities involved in the development process of an FPGA product, each with their contributions, incentives, and security risks. Naturally, we consider the same main parties as described in earlier works on FPGA design security [1], [13]–[15] and try to use consistent names and notations.

1) *The FPGA Vendor (FV):* develops and sells unconfigured FPGA devices, generally as off-the-shelve products.[2]
2) *The Core Vendor (CV):* offers access to its soft IP cores, i.e., innovative logical circuits for configuration on FPGAs, by licensing other parties to use them. We focus on pay-per-use licensing schemes with technical enforcement measures.
3) *The System Developer (SYS):* develops FPGA-based systems comprising a number of soft IP cores. The developed system can be an *end system* or an intermediate product to be embedded in another system. It is also in the interest of SYS to protect the FPGA bit stream of the end product in order to avoid end product cloning or reverse-engineering.
4) *The End User (EU):* pays for and uses the developed end system. He transforms the value of the product in money, which flows back to SYS, CV, FV according to their respective added value.
5) *A Trusted Third Party (TTP):* does not take part in the development process, but its goal is to create trust relationships between untrusting parties. The use of a TTP for such purpose is general practice in cryptographic infrastructures. The trust in a TTP is usually reputation based.

D. FPGA Protection Primitives

FPGA vendors already offer a number of increasingly stronger protection measures in earlier and more recent devices. We provide a short overview of the available primitives. Earlier proposed soft IP protection schemes based on these primitives and a discussion thereof can be found in Section III. Our newly proposed scheme is described in Section IV.

1) Device Identifiers: Most FPGA devices from all series and vendors have a public unique serial number burned-in at manufacturing which can be used as a device ID (e.g., Xilinx Device DNA [16]).

2) Nonvolatile FPGAs: A minor segment of the FPGA market consists of nonvolatile memory-based FPGAs which naturally offer a higher level of design security since the configuration data does not need to be externally stored, alleviating bus eavesdropping attacks. Vendors of SRAM-based FPGAs also offer more costly products emulating this behavior by tightly integrating a nonvolatile memory with a volatile FPGA (e.g., Xilinx Spartan-3 AN Series [17]). We will only consider purely volatile FPGAs in this work.

3) Bit Stream Integrity Checks: FPGA bit stream integrity is of the utmost importance, since faulty bit streams will cause erratic functionality and damage the FPGA. Cyclic redundancy checks or CRC codes are, therefore, an integral part of all bit stream formats. CRC works very well for random unintended errors in the bit stream, but is not adequate to detect intentional malicious changes. Some modern FPGA types offer more secure bit stream integrity checks (e.g., Xilinx Virtex-6 series [7]), using secure message authentication codes (MAC).

4) Decryption Support and Secure Key Storage: Many security problems with bit streams can be solved by encrypting them. Bit stream encryption offers similar protection as nonvolatile FPGAs, while only having to securely store a relatively short key. This is a typical security reduction achieved by using cryptographic primitives. Most volatile FPGA venors offer a hardwired on-chip decryption engine and secure key storage in their more high-end devices, which are solely accessible by the configuration controller and dedicated to bit stream decryption only. The corresponding encryption process is supported by the vendor's design tools, as are the methods to program a decryption key into the device. Once programmed, the key can never be read out externally and can internally only be accessed by the bit stream decryption engine.

5) Configuration Readback: Some FPGA types support configuration readback, i.e., the currently loaded bit stream can be read back over the FPGA's configuration pins, e.g., for debugging purposes. In general this worsens the FPGA design security problem since the FPGA configuration also needs protection

[1]One has to foresee that bit stream reversal tools could come available at any time in the near future, possibly under an open-source and hence low cost license.

[2]Designing the actual FPGA ICs also involves considerable engineering effort and the protection of the IP generated in these steps is equally important. In this work, we will not focus on the FPGA hardware itself, but on the IP protection of reconfigurable designs for FPGAs contained in bit streams, which we will call *soft IP*.

after it is loaded. Therefore, readback can be disabled explicitly and is for obvious reasons disabled automatically when bit stream encryption is used.

6) Partial (re)configuration: An important trend in recent FPGAs is partial configuration, i.e., to configure only a part of the reconfigurable array using partial bit streams. This offers a number of significant advantages, e.g., dedicated coprocessors can be configured when needed and the used FPGA logic can be released for other purposes afterwards. However, partial reconfiguration is known for causing a number of practical issues in older parts and tools and introducing manual design overhead. Luckily, recent development tools offer more and more automated support [7] and even run-time partial reconfiguration solutions become available [18]. Partial reconfiguration can also be disabled when undesirable, e.g., in combination with bit stream encryption.

7) Internal Configuration Access: In a number of recent FPGAs, the configuration controller can also be accessed internally by the FPGA's reconfigurable logic. For Xilinx FPGAs, this is done using the Internal Configuration Access Port (ICAP) primitive. ICAP can access all the functionality available to an external configuration port, including readback and partial reconfiguration. The combination of ICAP and partial reconfiguration leads to the particularly interesting technique of *self-reconfiguration*, i.e., the FPGA logic is able to reconfigure parts of itself. The ICAP primitive always has full access to the configuration controller, even when external access to readback and partial reconfiguration is disabled.

III. Related Work

A number of solutions for (modular) IP protection on ASIC devices have been proposed [19]–[23], but these are not directly applicable to FPGA systems and are not discussed in more detail here. Instead, in this section, we focus on earlier proposed FPGA design security schemes. We first briefly discuss the schemes proposed by the FPGA vendors and subsequently other related research. Finally, we summarize the position of this work's contribution.

A. Commercial Proposals

1) IFF Copy Protection: A relatively simple proposed scheme to protect FPGA bit streams against direct cloning makes use of a so-called *Identification Friend or Foe (IFF)* scheme [24]–[26]. The key idea of the IFF scheme is to accompany every FPGA with an external secure device bearing a unique secret key. The FPGA checks the presence of its companion before enabling its functionality. In this way, direct cloning is prevented since the cloner, not knowing the secret key, cannot clone the secure device. In practice, the external device is a secure EEPROM implementing a keyed hash function and a one-time-programmable secure key storage. The same key is embedded in the FPGA's bit stream and the FPGA checks the presence of the secure device by means of a challenge-response protocol with a random nonce. If the secure device responds correctly, it is identified as a friend and the FPGA is enabled, otherwise it is a foe and the FPGA stops functioning. If the used key is secret and unique, only the genuine secure EEPROM can calculate the expected response, and a failure to do so indicates a cloned design. A number of issues can be pointed out about this very simple scheme, the most critical being the embedding of a secret key in the plaintext bit stream. We mentioned before that the *obfuscation* offered by the bit stream's encoding is not sufficient to securely store secret data like cryptographic keys.

2) Device ID Checking: An alternative simple scheme to protect bit streams against direct cloning involves the unique identifier burned in every FPGA. The identifier is read and processed by a "security algorithm" into a check code. This check code is compared to an externally stored reference code and the design is only enabled if both codes match. Since the device identifier is publicly accessible, the security of this scheme is based on a secret element in the security algorithm, e.g., a keyed hash function with the secret key embedded in the bit stream. Simply cloning the bit stream and using it on another FPGA will not work since the device identifier is different and the calculated check code does not match the stored reference code. In [27], Xilinx proposes a number of variants of this scheme using the device DNA and additional nonvolatile public identifier strings. The same security issue as in the IFF proposals still hold, i.e., a secret is embedded in the bit stream and hence not really protected.

3) Bit Stream Integrity Checking: A simple method to detect tampering during operation of an FPGA design makes use of internal configuration readback (see Section II-D). Using ICAP, an FPGA design reads back its current configuration and calculates a short check code on this data. This check code is compared with an externally stored reference check code to detect whether (malicious) tampering occurred. In [27], a practical implementation is proposed using a CRC code as a check. We pointed out that CRC codes are not sufficient to provide protection against malicious tampering and, therefore, it is highly recommended to use a secure cryptographic integrity check instead. A second issue here is that the reference check code is not authenticated. An adversary can tamper with the bit stream and alter the reference check accordingly to pass the integrity test.

4) Bit Stream Encryption: Most SRAM-based FPGA vendors provide bit stream encryption support for their high-end devices, using standardized secure encryption algorithms, and options to store a nonvolatile decryption key. Xilinx devices support bit stream encryption [7], [28] starting from the Virtex-II series and store the decryption key in a dedicated battery-backed SRAM or, starting from their Spartan/Virtex-6 series, in a one-time programmable eFuse register. Xilinx Virtex-6 FPGAs moreover support *authenticated encryption* [29] of bit streams by also providing a dedicated secure HMAC implementation. Altera devices offer bit stream encryption [30], [31] starting from the Stratix-II series and developers have the choice between battery-backed SRAM and one-time programmable polyfuses to store the decryption keys. We note that very recently a successful side-channel attack was found for the bit stream decryption engines used on Xilinx Virtex-II [32] and Virtex-4/5 FPGAs [33]. These results painfully highlight the need for securely implemented cryptographic building blocks on FPGA devices, since the successful deployment of any secure soft IP protection scheme, including the one presented in this work, rests on the availability of such primitives.

B. Academic Research

Modern FPGA applications often implement a microcontroller running software in combination with application specific coprocessors. The protection of this software's IP (SW-IP) from cloning is very similar to soft IP protection discussed earlier. In [14], Simpson *et al.* introduce an authentication scheme for SW-IP in reconfigurable systems like FPGAs. Their proposal is a more lightweight alternative to Trusted Computing [10], which is used in typical PCs for this purpose, and succeeds both in protecting the IP rights of the SW-IP provider and assuring the authenticity of the software to the system developer. The implementation by Simpson *et al.* makes use of a Physically Unclonable Function or PUF [34], a hardware primitive able to produce an unclonable device fingerprint. In [14], the PUF is used as a combined unique device identifier and secure key generator which alleviates the need for protected on-board nonvolatile key storage. Besides a PUF, Simpson *et al.* also propose to use a TTP for authenticated SW-IP distribution. Variants of this scheme are presented by Guajardo *et al.* with additional security notions [15], [35]. Guajardo *et al.* also propose concrete and practical FPGA-based PUF constructions, whereas Simpson *et al.* merely assumed the existence of a secure and reliable PUF on the FPGA.

Gora *et al.* in [36] present an alternative way for protecting SW-IP in FPGAs by binding it to a hardware platform using a PUF in the FPGA's reconfigurable logic. Their proposed method specifically protects software and assumes that the hardware soft IP cores are securely configured by other means of protection. The scheme requires the secure embedding of a hash value in the bit stream for integrity checking. They acknowledge the possibility of bit stream reversal and investigate the safest method to hide a value in the bit stream format. They propose to use the bit stream's routing information since this is presumably the hardest to reverse accurately. However, this embedding cannot be assumed to be of a cryptographically acceptable security level.

In [37], Bossuet *et al.* propose a solution which offers the flexibility to provide a granular protection level of different parts of the bit stream, based on the security-critical aspect of the considered design modules. This is achieved by using the partial configuration and internal configuration access possibilities of Xilinx FPGAs, as discussed in Section II-D. However, the proposed scheme does not distinguish between soft IP providers (CV) and system developers (SYS). The system developer knows all plaintext bit streams and hence has full access to all implementation details. The proposal only provides protection against cloning or reverse-engineering of a complete FPGA configuration, but does not protect individual soft IP cores.

Güneysu *et al.* [1] propose a volume licensing scheme for FPGA bit streams which requires only small changes to the configuration controller, i.c., a secondary secure key register and the use of *authenticated* bit stream encryption. Their proposal points out the lack of a convenient key transport and installation scheme with the available bit stream encryption options, which prevents flexible protection methods. They solve this problem by using a public-key-based key agreement protocol between the soft IP provider and the FPGA and also make use of a trusted third party, i.c., the FPGA vendor. The solution is limited to the protection of full FPGA configurations and the protection of partial soft IP cores is labeled as a significant open problem. Drimer *et al.* [2] discuss a possible extension to this proposal for the protection of multiple cores. Our proposal is closely related to the work of Güneysu *et al.* and Drimer *et al.* but we aim to solve some of the open problems and provide a more practical implementation. Therefore, we provide a comparative analysis between both in Section V-E, after we have introduced our scheme.

TABLE I
OVERVIEW OF DIFFERENT FPGA DESIGN SECURITY SOLUTIONS

Proposal	Intended for soft IP (HW)	Feasible in current devices	Cryptographically secure	Protection for SYS's and CV's IP
IFF, Device ID checking	√	√	×	×
Bitstream encryption	√	√	√	×
Simpson et al. [14]	×	×	√	√
Guajardo et al. [15], [35]	√	×	√	√
Gora et al. [36]	×	√	×	×
Bossuet et al. [37]	√	×	√	×
Güneysu et al. [1]	√	×	√	√
Drimer et al. [2]	√	×	√	√
Our scheme (Sect. IV)	√	√	√	√

C. Position of Our Contribution

The contribution presented in this work is a flexible and cryptographically secure scheme for the protection of soft IP material of both SYS and CV, against cloning and reverse-engineering by low class and middle class adversaries. Moreover, the proposed scheme is feasible in recent commercially available devices. This is in contrast to the proposals from [1], [2], [14], [15], [35], and [37] which all require modifications to the FPGA hardware such as PUFs, extra key registers, or additional cryptographic primitives, none of which are currently supported by any FPGA vendor.

The commercial proposals are obviously achievable using current devices but suffer from a number of security-related weaknesses that have been discussed in Section III-A. The only commercial solution which offers cryptographic security is bit stream encryption, but in its current form has some practical drawbacks related to key management as also pointed out in [1] and [38]. Moreover, bit stream encryption only secures monolithic bit streams and does not protect against unauthorized use of individual soft IP blocks, e.g., by a system developer. As summarized in Table I, none of the discussed earlier solutions are both secure and feasible in current devices, as well as flexible enough to offer modular soft IP protection. The scheme as proposed in this work aims to close this gap.

IV. Flexible Active Metering Scheme

A. Preliminary

Notation: By b(IP) we mean the (partial) bit stream representation of a soft IP core design IP. Encryption of a message m using a key k is denoted as $M := \texttt{Enc}[m]_k$ and the corresponding decryption as $m := \texttt{Dec}[M]_k$. We use B(IP) to denote encrypted bit streams: $\mathrm{B(IP)} := \texttt{Enc}[\mathrm{b(IP)}]_k$ and K to denote encrypted keys: $K := \texttt{Enc}[k]_{k'}$. F symbolizes an FPGA device. Identifier values denoted as $\texttt{ID}(F_i)$ or $\texttt{ID}(\mathrm{IP}_j)$ are used to uniquely refer to a particular FPGA or soft IP core.

1) Device Capabilities: For our scheme, we consider FPGA devices F with the following capabilities:

- F is uniquely identifiable by means of a public identifier $\texttt{ID}(F)$. This can be, e.g., a printed serial number or an embedded unique bitstring.
- F supports bit stream decryption by means of a securely implemented and dedicated on-chip engine which can only be accessed by the configuration controller. It is also assumed that external (re)configuration and configuration readback are disabled.
- F is able to store a bit stream decryption key for the on-chip decryption engine in a secure nonvolatile memory. We distinguish between *blank* devices F^* where no key has been programmed yet and enrolled devices F with a programmed nonvolatile key. The act of programming a nonvolatile decryption key k in a blank FPGA device F^* is denoted as $F := \left[F^* \overset{\text{NVM}}{\longleftarrow} k\right]$.
- F supports partial bit stream (re)configuration.
- F is able to internally access its configuration controller through an ICAP primitive.

We remark that currently FPGA devices are for sale which support all these capabilities, notably Xilinx' Virtex-6 [7] series which support AES-256 bit stream decryption, are able to store a 256-bit AES key in eFuses or battery-backed SRAM and support partial reconfiguration and ICAP primitives.

B. Metering Authority (MA)

The proposed scheme is based on a TTP which provides a metering service for soft IP. From here on we will refer to this party as the *metering authority (MA)*. MA plays a central role in the scheme and is explicitly trusted by all other parties to run a correct and secure service.

1) Role of the Metering Authority: The key idea of the proposed scheme is that MA acts as a trusted party, linking FPGA devices and soft IP cores. The protocol which implements this idea is shown in Fig. 1 and described in detail in Section IV-C. It consists of three main parts:

1) FVs enroll their FPGA devices with MA [Fig. 1(a)].
2) CVs enroll their soft IP cores with MA [Fig. 1(b)].
3) SYS (or EU) interacts with MA in order to obtain a license for the activation of a particular soft IP core on a particular FPGA device [Fig. 1(c)].

Instances of a soft IP core are activated on a per-device basis and obtaining an activation license requires an explicit contact with MA. MA can hence keep track of exactly how many instances of the IP core are activated. This is called *active metering* of the IP core.

2) Metering Bit Stream $\mathbf{b}(M, k_i^M)$*:* A compact custom metering design is required in order to bootstrap the secure configuration of protected soft IP cores on enrolled FPGA devices. The metering bit stream $\mathbf{b}(M, k_i^M)$, provided by MA, implements the following modules, as shown in Fig. 2:

a) A custom decryption module implementing a secure symmetric cipher.[3] Note that the on-chip bit stream decryption engine provided by the FPGA vendor cannot be reused in the metering bit stream, since it does not allow us to use an alternative key and it is not directly accessible from the FPGA's configurable logic.
b) Two key registers for the decryption module. The first one is called the *metering key register* and is preloaded with a device-unique metering key k_i^M which is embedded in the metering bit stream by MA. We do not assume any obfuscation properties of the metering bit stream with respect to the embedded metering key. Instead, we will make sure that the metering bit stream is only transferred in encrypted form to protect the metering key explicitly. The second key register is the *IP key register* which is empty upon initial configuration but is loaded with an appropriate IP key during the activation sequence of a protected IP core as shown in Fig. 4.
c) An ICAP interface, which is usually an instantiation of a custom library primitive provided by the FPGA vendor.

C. Description of the Protocol

1) FPGA Device Enrollment: To enable the proposed scheme, MA has to have physical access to the FPGA devices once in a secure environment; this is called *device enrollment*. For every device F_i which is enrolled, MA has to do two things:

a) MA programs a secret bit stream decryption key k_i^F in the dedicated and secured nonvolatile key register in the device. This key is called the *device key* and is unique for every device.
b) MA provides each enrolled device with a unique encrypted metering bit stream $\mathbf{B}_i(M, k_i^M) := \texttt{Enc}[\mathbf{b}(M, k_i^M)]_{k_i^F}$. The *metering bit stream* implements the metering design M and embeds a unique *metering key* k_i^M.

To prevent unenrollment of devices, k_i^F can be stored in one-time programmable memory. $\mathbf{B}_i(M, k_i^M)$ is stored in external nonvolatile memory and it is automatically decrypted and configured on F_i immediately after device power-up. Note that failure to retain k_i^F or to configure $\mathbf{b}(M, k_i^M)$ does not break the scheme's security, but merely disables F_i to load any soft IP which is actively metered by MA. MA keeps track of the devices and the corresponding device and metering keys which it has enrolled, by maintaining a device database with entries $\{\texttt{ID}(F_i), k_i^F, k_i^M\}$. The device enrollment protocol is shown in Fig. 1(a). Note that the device key k_i^F is not explicitly needed in the metering scheme anymore after device enrollment, but it is stored by MA for service reasons, e.g., to provide future updates of the metering bit stream.

[3]To completely protect the integrity of a design, e.g., as a countermeasure for bit stream tampering, the use of a decryption module which supports authenticated decryption is recommended.

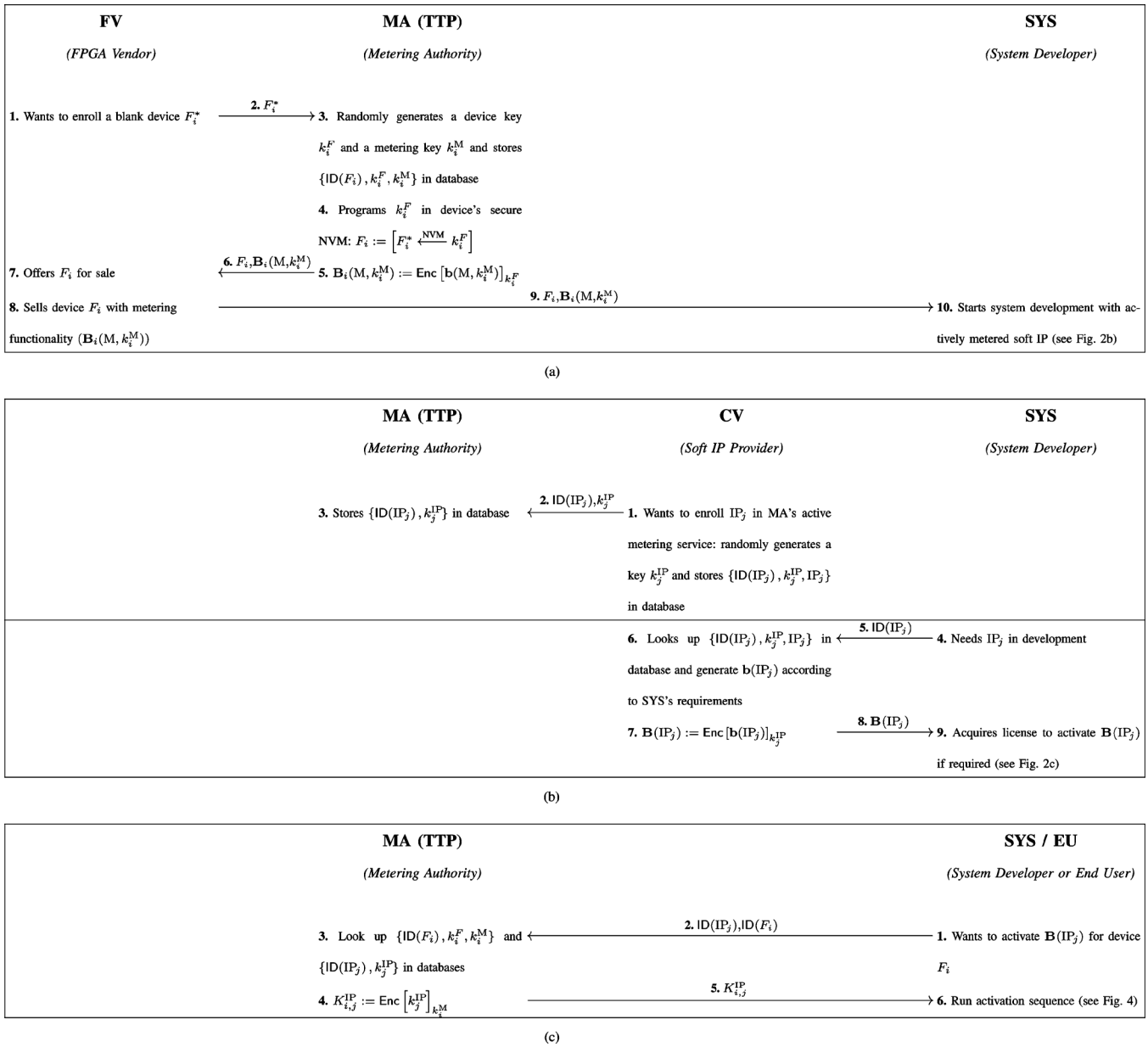

Fig. 1. Soft IP core active metering protocol. (a) FV enrolls devices with MA and sells devices to SYS. (b) CV enrolls soft IP with MA and distributes soft IP to SYS. (c) MA licenses soft IP to SYS or EU.

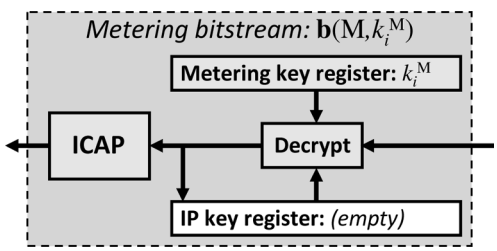

Fig. 2. Modules implemented by the metering bit stream $\mathbf{b}(M, k_i^M)$.

2) Soft IP Core Enrollment: A CV who wants to enroll a soft IP core IP_j in the active metering service offered by MA has to pick a unique and secure encryption key k_j^{IP} linked to this piece of soft IP. k_j^{IP} is called the *IP key* and is communicated to the MA over a secured channel together with a unique reference for this IP core: $\mathtt{ID}(\mathrm{IP}_j)$. MA registers the enrolled soft IP core as an entry $\{\mathtt{ID}(\mathrm{IP}_j), k_j^{\mathrm{IP}}\}$ in its IP database. Now, when CV receives a request from SYS to obtain the soft IP core IP_j, CV sends a protected version of the bit stream of IP_j by encrypting it with the corresponding IP key: $\mathbf{B}_{(\mathrm{IP}_j)} := \mathtt{Enc}[\mathbf{b}(\mathrm{IP}_j)]_{k_j^{\mathrm{IP}}}$. This protocol is shown in Fig. 1(b).[4] SYS cannot directly integrate the obtained IP in its system since he cannot decrypt it. We say $\mathbf{B}_{(\mathrm{IP}_j)}$ needs a *license* to be activated.

3) Soft IP Core Licensing: It is clear that SYS needs a license containing the correct IP key k_j^{IP} in order to activate the protected bit stream $\mathbf{B}_{(\mathrm{IP}_j)}$. However, SYS should not be able to see k_j^{IP} directly, since that would allow him to obtain the

[4]Note that in practice, the bit stream $\mathbf{b}(\mathrm{IP}_j)$ needs to meet a number of customer specific requirements; e.g., detailing the size, location, and interface of the IP core in the customer's system. Therefore, $\mathbf{b}(\mathrm{IP}_j)$ is generated upon request for a particular SYS.

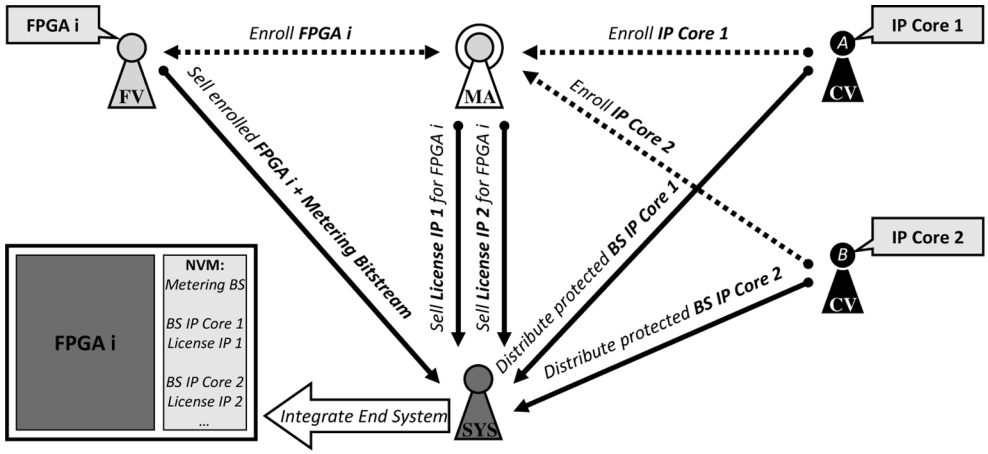

Fig. 3. Simplified overview of communications between involved parties. In the example shown, SYS produces an FPGA system containing two soft IP core bit streams (BS) from different core vendors.

plain text bit stream $\mathbf{b}(\mathrm{IP}_j)$ and circumvent the active metering control. Therefore, k_j^{IP} is encrypted with the metering key of F_i: $K_{i,j}^{\mathrm{IP}} := \mathtt{Enc}[k_j^{\mathrm{IP}}]_{k_i^M}$. This encrypted IP key $K_{i,j}^{\mathrm{IP}}$ serves as the *license* to activate the protected soft IP core. Since it is still encrypted, SYS does not learn the actual value of k_j^{IP}. The licensing protocol is shown in Fig. 1(c). The only party able to generate valid licenses is MA since he holds the database of both enrolled devices, containing $\mathtt{ID}(F_i)$ and the corresponding metering key k_i^M, as well as a database of the enrolled soft IP cores, containing $\mathtt{ID}(\mathrm{IP}_j)$ and the corresponding IP key k_j^{IP}.

D. System Integration and Soft IP Core Activation

To develop an FPGA-based product, a system developer obtains an FPGA device with accompanying metering bit stream from an FPGA vendor [following Fig. 1(a)], the required third-party soft IP cores from core vendors [following Fig. 1(b)] and the required licenses for these cores from the metering authority [following Fig. 1(b)]. A simplified overview of all these communications between the involved parties is shown in Fig. 3 for an example where two different IP cores are obtained from two different core vendors. Once SYS has all these elements, he is able to integrate them in the end system by putting the metering bit stream, the protected IP core bit streams, and their accompanying licenses in a nonvolatile memory next to the FPGA device. When the system is powered on, the activation sequence, as shown in Fig. 4, loads the protected cores in the reconfigurable FPGA fabric. First the metering bit stream containing the metering key k_i^M is loaded [Fig. 4(a)]. In the second step, the license $k_{i,1}^{\mathrm{IP}}$ is decrypted to the IP key k_1^{IP} which is temporarily loaded in the IP key register [Fig. 4(b)]. Next, the first protected bit stream $\mathbf{B}(\mathrm{IP}_1)$ is loaded directly into the metering logic (not to the configuration controller). The metering bit stream decrypts $\mathbf{B}(\mathrm{IP}_1)$ to $\mathbf{b}(\mathrm{IP}_1)$ with the loaded IP key k_1^{IP}. Finally, the decrypted soft IP core bit stream $\mathbf{b}(\mathrm{IP}_1)$ is sent to the configuration controller, using the internal ICAP interface, and configured in the designated location of the reconfigurable logic array [Fig. 4(c)]. The activation scheme is repeated for every individual IP core for which a license is available, i.c., for IP core 2 [Fig. 4(d) and (e)].

V. Discussion of the Proposed Metering Scheme

A. Security Evaluation of the Scheme

When the metering scheme, as shown in Fig. 1, and the activation sequence, as shown in Fig. 4, are executed correctly, the considered soft IP core is protected from cloning and reverse-engineering by low and middle class adversaries. All data transferred between parties in Fig. 1, and all data stored externally to the FPGA device in Fig. 4 are encrypted using secure algorithms and keys.[5] All security sensitive operations are performed either in a protected environment, e.g., at the site of MA or CV, or inside the FPGA device. Since low and middle class adversaries are not able to gain physical access to the FPGA's internals, the protected bit streams remain secret. Since IP core licenses $K_{i,j}^{\mathrm{IP}}$ are unique to a particular device/IP core pair $\{F_i, \mathrm{IP}_j\}$, there is no advantage in duplicating them for use with another device and/or IP core.

The scheme does not protect against high class adversaries since they may be able to extract a device key or metering key by probing the FPGA internals. This is a costly attack, but since it allows the adversary to decrypt former and even future protected bit streams, it might be worthwhile. There is not much to be done about the loss of protection of former bit streams when a metering key is leaked. However, when a disclosed metering key is detected,[6] MA is at least able to protect future soft IP cores by revoking the particular metering key; i.e., MA will not grant licenses for this device any longer. In order to decrypt future protected bit streams, the adversary needs to rerun the costly attack on a different FPGA. By revoking broken metering keys, partial protection against high class adversaries is obtained.

Finally, we note that the integrity of the metering bit stream in our scheme is not strictly protected. An adversary might hence alter the metering bit stream; however, he does not learn the contents. To offer complete security against cases where an adversary is able to make meaningful alterations to the encrypted metering bit stream, authenticated encryption

[5]For the communication of the IP key from CV to MA [step 2 in Fig. 1(b)], we assume that the channel is secured using conventional cryptographic methods.

[6]We will not go into detail concerning cloning detection methods such as IP watermarking; see, e.g., [39] for a survey.

needs to be used. Currently, Xilinx Virtex-6 devices support authenticated encryption by implementing an HMAC based on SHA-256. However, even if there is no on-board support for bit stream authentication, it is still possible to protect against design alterations by having a design check its own integrity using bit stream readback, as detailed in Section III-A3.

B. Practical Issues

1) Partial Reconfiguration Issues: In this work, we assume an ideal scenario where all the practical details concerning partial reconfiguration are transparent to the involved parties. In reality, this is unfortunately not (yet) the case. Using partial reconfiguration will produce nonnegligible overhead, both in design effort as well as silicon resources. As the capabilities of FPGA development tools progress, it is expected that the impact of this overhead will decrease up to the point where it does become transparent. For now, however, it needs to be taken into account when implementing the scheme in practice. The practical implications of using partial reconfiguration have no effect on the security or applicability of the proposed scheme and are outside the scope of this work.

2) Development Flow and Simulation Issues: IP protection methods for hardware cores in general often stand in the way of common development flows, in particular an encrypted form of a core's description restricts simulation, verification, or offline testing. Solutions to this problem are based on encrypted netlists and the availability of trusted design tools and are hence in the scope of secure software research. The development and application of trusted design tools should be considered orthogonal to this work. Using standardized bus interfaces and protocols and providing detailed core specifications will also relax the needs for offline simulation and verification. We note that a core vendor is able to allow online simulation by providing a number of core licenses at no cost for simulation purposes. There is no risk of IP abuse in this case since these licenses are device-locked.

C. Variants of the Scheme

1) End User Activation: The licensing and activation of a protected soft IP core can be done by the end user EU as well. SYS has the possibility to embed a protected, but not yet activated soft IP core, e.g., as an extra feature of the product. If EU wishes to use this feature, he has to acquire the license from MA and run the activation himself.

2) Master Keys Instead of Key Databases: At a number of instances in the proposed scheme, keys are randomly generated and linked to a corresponding device or IP core by means of storing and retrieving them from a database indexed by an identifier [step 3 in Fig. 1(a), steps 1, 3, and 7 in Fig. 1(b), and step 3 in Fig. 1(c)]. In these steps, the use of database storage can be replaced by a secure deterministic key derivation algorithm based on the identifier and a secret master key from the considered party. A secure keyed MAC algorithm would be an appropriate choice, e.g., in step 3 in Fig. 1(a): $k_i^M := \mathrm{MAC}\left[\mathrm{ID}(F_i)\right]_{\mathbf{k}^{\mathrm{MA}}}$, with $\mathbf{k}^{\mathrm{MA}}$ MA's metering master key. When MA requires a particular device's metering key later on, e.g., in step 3 in Fig. 1(c), he can regenerate it using his metering master key and the key

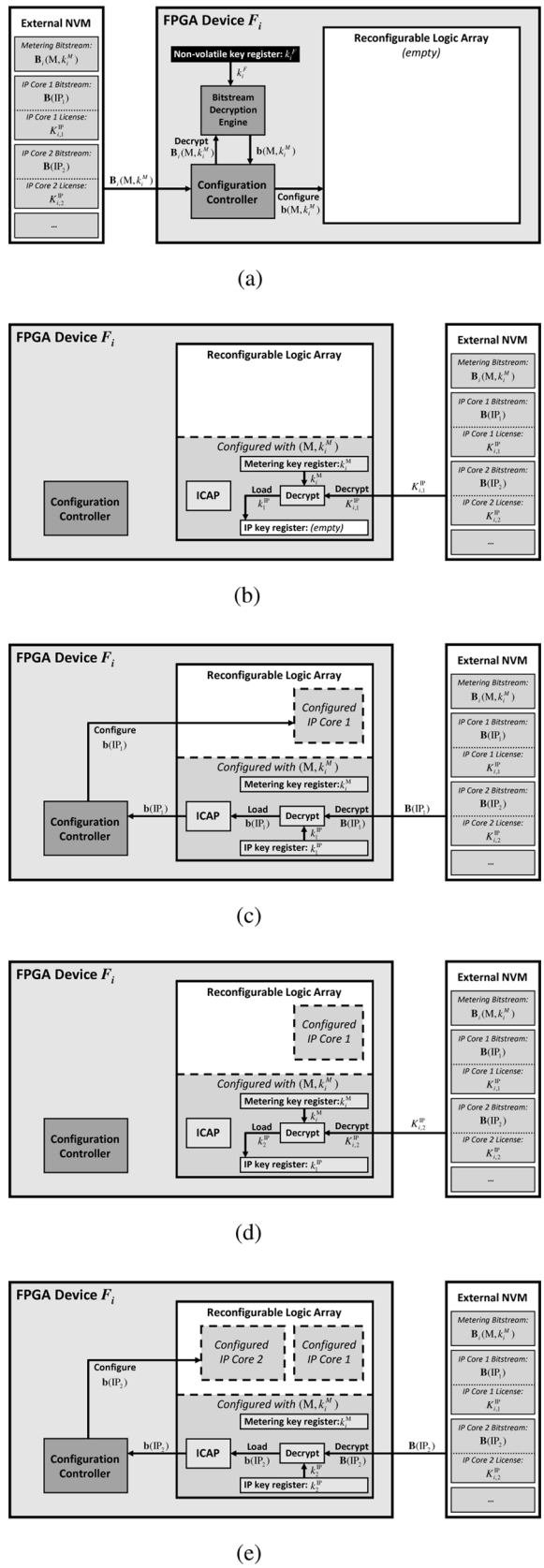

Fig. 4. Activation sequence of an FPGA system containing multiple protected soft IP cores. (a) Initialize metering: load the metering bit stream $b(M, k_i^M)$. (b) License IP Core 1: load IP Core 1 decryption key k_i^{IP}. (c) Configure IP Core 1: load protected IP Core 1 bit stream B (IP_1). (d) License IP Core 2: load IP Core 2 decryption key k_2^{IP}. (e) Configure IP Core 2: load protected IP Core 2 bit stream B (IP_2).

derivation algorithm, lifting the need for a metering key database. In case of IP keys, a secret IP master key needs to be securely shared between CV and MA. Note that the latter variant also reduces the online communication overhead between CV and MA, i.e., step 2 in Fig. 1(b) is not necessary anymore and the soft IP core enrollment is done implicitly. The security of the used key derivation algorithm and the secrecy of the master keys are in this variant of great importance.

3) Reducing Back-End Load for CV: In the current scheme, CV also maintains a database of IP cores IP_j and generates protected IP core bit streams $\mathbf{B}(\mathrm{IP}_j)$ upon request. This allows him to keep full control over the bit stream, e.g., he can issue updates to IP_j without having to recontact MA, as long as he uses the same IP key. However, maintaining such a database service might be an expensive back-end load for a small CV. An alternative would be that CV encrypts $\mathbf{b}(\mathrm{IP}_j)$ once, registers the IP key with MA and publishes $\mathbf{B}(\mathrm{IP}_j)$, e.g., on a website. In that case, CV does not have to keep an IP core nor an IP key database. Parties which wish to use the IP core can download it publicly and acquire a license from MA. This variant minimizes the back-end service for CV but also reduces the flexibility since updates have to be made explicit and no SYS-specific requirements can be included.

D. Performance Evaluation

At the device side, the overhead of using the proposed scheme is minimal. The only significant addition is the metering bit stream. The resource usage of the metering bit stream, as shown in Fig. 2, is very low since it only implements two key registers and a decryption module.[7] The size of the decryption module depends on the algorithm used, but secure and efficient implementations of block ciphers exist for FPGAs [40], [41]. All together, the metering bit stream can be implemented occupying only a marginal fraction of the reconfigurable resources on recent high-end FPGAs. At the back-end side, i.e., for the MA and CV, the overhead mainly consists of secure database management, and this can even be substantially reduced according to the suggested variants proposed in Section V-C.

E. Comparison to Previous Proposals

As a final discussion topic, we compare our scheme to similar previously proposed soft IP protection schemes, in particular the work of Güneysu *et al.* [1] and the extension for multiple cores as proposed by Drimer *et al.* [2]. These schemes focus on the same problem and are also based on the use of a TTP and partial reconfiguration techniques. A major difference is that these schemes make use of public key cryptography for key distribution which typically introduces a much larger implementation overhead than the symmetric primitives used in our scheme. The proposed ECDH core in [1] requires 2706 slices whereas an AES block cipher which is sufficient to support our scheme can be implemented in as few as 124 slices [40]. The use of public key primitives does simplify the required interactions in their protocols and they only need a TTP during the enrollment phase and for revocation. However, the transaction size of their interactions during the distribution phase can become incredibly large ("multiple gigabytes" [2]) since a different encrypted IP core is required for every FPGA device. In our scheme, the interactions are slightly more complex, but the size of the communications is kept minimal since the same protected bit stream is used for every FPGA device. Only the licenses are unique but these are very small. Finally, as pointed out in Section III-C, the previous schemes do require (albeit small) changes to the FPGA hardware in the form of additional key registers, whereas our scheme can be implemented on existing devices without any hardware modifications.

[7]The ICAP primitive is a hard-wired element and does not occupy any reconfigurable logic primitives.

VI. Conclusion

In this work, we have proposed an active metering scheme for the protection of FPGA configurations at the level of individual IP cores and argued why such a scheme is indispensable in a system-level development model for modern FPGAs. The use of an *active* scheme allows the IP provider to implement a pay-per-use licensing model. The proposed solution is, moreover, the first known construction for this level of IP protection which is realizable in existing FPGA devices.

References

[1] T. Güneysu, B. Moller, and C. Paar, "Dynamic intellectual property protection for reconfigurable devices," in *Proc. Int. Conf. Field Programmable Technology (ICFPT)*, 2007, pp. 169–176.

[2] S. Drimer, T. Güneysu, M. G. Kuhn, and C. Paar, Protecting Multiple Cores in a Single FPGA Design 2008 [Online]. Available: http://www.cl.cam.ac.uk/~sd410/papers/protect_many_cores.pdf.

[3] Intel atom processor E6x5C series Intel Product Preview Datasheet, 324602-001US, Dec. 2010.

[4] Xilinx redefines power, performance, and design productivity with three new 28 nm FPGA families: Virtex-7, kintex-7, and artix-7 devices (v1.0) Xilinx White Paper 373, Jun. 2010.

[5] Stratix V FGPAS: Built for bandwidth Altera Brochure, 2010.

[6] AXI4 interconnect paves the way to plug-and-Play IP (v1.0) Xilinx White Paper 379, Oct. 2010.

[7] Xilinx Virtex 6 Configuration User Guide (v3.2) 2010.

[8] J. B. Note and E. Rannaud, "From the bitstream to the netlist," in *Proc. ACM/SIGDA Symp. Field-Programmable Gate Arrays (FPGA)*, Nov. 2008, pp. 264–264.

[9] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Des. Test Comput.*, to be published.

[10] D. Grawrock, *Dynamics of a Trusted Platform: A Building Block Approach*, 1st ed. U.S.: Intel Press, 2009.

[11] D. G. Abraham, G. M. Dolan, G. P. Double, and J. V. Stevens, "Transaction security system," *IBM Syst. J.*, vol. 30, pp. 206–229, Mar. 1991.

[12] R. Torrance and D. James, "The state-of-the-Art in IC reverse engineering," in *Proc. Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2009, pp. 363–381.

[13] T. Kean, "Cryptographic rights management of FPGA intellectual property cores," in *Proc. ACM/SIGDA Symp. Field-Programmable Gate Arrays (FPGA '02)*, 2002, pp. 113–118.

[14] E. Simpson and P. Schaumont, "Offline hardware/software authentication for reconfigurable platforms," in *Proc. Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2006, pp. 311–323.

[15] J. Guajardo, S. S. Kumar, G. J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in *Proc. Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2007, pp. 63–80.

[16] Security solutions using Spartan-3 generation FPGAs (v1.1) Xilinx White Paper 266, Apr. 2008.

[17] Spartan-3 AN FPGA family data sheet (v4.0) Xilinx Data Sheet 557, Dec. 2010.

[18] D. Koch, C. Beckhoff, and J. Teich, "ReCoBus-builder—A novel tool and technique to build statically and dynamically reconfigurable systems for FPGAs," in *Proc. Int. Conf. Field-Programmable Logic and Applications (FPL 08)*, Heidelberg, Germany, Sep. 2008, pp. 119–124.

[19] Y. Alkabani, F. Koushanfar, and M. Potkonjak, "Remote activation of ICs for piracy prevention and digital right management," in *Proc. Int. Conf. Computer-Aided Design (ICCAD)*, 2007, pp. 674–677.
[20] Y. Alkabani and F. Koushanfar, "Active hardware metering for intellectual property protection and security," in *Proc. USENIX Security Symp.*, 2007, pp. 291–306.
[21] J. A. Roy, F. Koushanfar, and I. L. Markov, "EPIC: Ending piracy of integrated circuits," in *Proc. Design Automation and Test in Europe (DATE)*, 2008, pp. 1069–1074.
[22] J. A. Roy, F. Koushanfar, and I. L. Markov, "Protecting bus-based hardware IP by secret sharing," in *Proc. Design Automation Conf. (DAC)*, 2008, pp. 846–851.
[23] R. Maes, D. Schellekens, P. Tuyls, and I. Verbauwhede, "Analysis and design of active IC metering schemes," in *Proc. IEEE Workshop on Hardware-Oriented Security and Trust (HOST)*, 2009, pp. 74–81.
[24] FPGA IFF copy protection using dallas semiconductor/maxim DS2432 secure EEPROMS (v1.1) Xilinx App. Note 780, May 2010.
[25] An FPGA design security solution using a secure memory device (v1.0) Altera White Paper 01033, Oct. 2007.
[26] FPGA design security solution using MAX II devices (v1.0) Altera White Paper M2DSGN, Sep. 2004.
[27] Advanced security schemes for Spartan-3 A/3 AN/3 A DSP FPGAs (v1.0) Xilinx White Paper 267, Aug. 2005.
[28] Using high security features in Virtex-II series FPGAs (v1.0) Xilinx App. Note 766, Jul. 2004.
[29] S. Trimberger, J. Moore, and W. Lu, "Authenticated encryption for FPGA bitstreams," in *Proc. ACM/SIGDA Symp. Field-Programmable Gate Arrays (FPGA)*, 2011, pp. 83–86.
[30] Protecting the FPGA design from common threats (v1.0) Altera White Paper 01111, Jun. 2009.
[31] Design security in Stratix III devices (v1.5) Altera White Paper 01010, Sep. 2009.
[32] A. Moradi, A. Barenghi, T. Kasper, and C. Paar, On the Vulnerability of FPGA Bitstream Encryption Against Power Analysis Attacks—Extracting Keys From Xilinx Virtex-II FPGAs Cryptology ePrint Archive, Rep. 2011/390, 2011.
[33] A. Moradi, M. Kasper, and C. Paar, On the Portability of Side-Channel Attacks—An Analysis of the Xilinx Virtex 4 and Virtex 5 Bitstream Encryption Mechanism Cryptology ePrint Archive, Rep. 2011/391, 2011.
[34] R. Maes and I. Verbauwhede, "Physically unclonable functions: A Study on the state of the art and future research directions," in *Towards Hardware-Intrinsic Security*, ser. Security and Cryptology, D. Naccache and A. R. Sadeghi, Eds. New York: Springer, 2010, pp. 3–37.
[35] S. S. Kumar, J. Guajardo, R. Maes, G. J. Schrijen, and P. Tuyls, "The butterfly PUF: Protecting IP on every FPGA," in *Proc. IEEE Workshop on Hardware-Oriented Security and Trust (HOST)*, 2008, pp. 67–70.
[36] M. Gora, A. Maiti, and P. Schaumont, "A flexible design flow for software IP binding in commodity FPGA," in *Proc. IEEE Symp. Industrial Embedded Systems (SIES)*, Jul. 2009, pp. 211–218.
[37] L. Bossuet, G. Gogniat, and W. Burleson, "Dynamically configurable security for SRAM FPGA bitstreams," *Int. J. Eng. Sci.*, vol. 2, no. 1/2, pp. 73–85, 2006.
[38] S. Drimer, "Security for Volatile FPGAs," Ph.D., Univ. of Cambridge, Cambridge, 2009.
[39] A. T. Abdel-Hamid, S. Tahar, and E. M. Aboulhamid, "A survey on IP watermarking techniques," *Des. Autom. Embedded Syst.*, vol. 9, pp. 211–227, 2004.
[40] T. Good and M. Benaissa, "AES on FPGA from the fastest to the smallest," in *Proc. Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2005, pp. 427–440.
[41] M. Sbeiti, M. Silbermann, A. Poschmann, and C. Paar, "Design space exploration of PRESENT implementations for FPGAs," in *Proc. Southern Conf. Programmable Logic (SPL)*, 2009, pp. 141–145.

Roel Maes (S'11) received the electrical engineering degree from the Katholieke Universiteit Leuven (K.U.Leuven), Belgium, in 2007. He is currently working toward the Ph.D. degree at the Computer Security and Industrial Cryptography (COSIC) Laboratory, Electrical Engineering Department (ESAT), K.U.Leuven.

His research interests include physically unclonable functions (PUFs) and hardware design security.

Dries Schellekens received the electrical engineering degree from the Katholieke Universiteit Leuven (K.U.Leuven), Belgium, in 2002. He is currently working toward the Ph.D. degree at the Computer Security and Industrial Cryptography (COSIC) Laboratory, Electrical Engineering Department (ESAT), K.U.Leuven.

His research interests include trusted computing and hardware design security.

Ingrid Verbauwhede (M'92–SM'00) received the electrical engineering degree and the Ph.D. degree from the Katholieke Universiteit Leuven (K.U.Leuven), Belgium, in 1991.

From 1992 to 1994, she was a postdoctoral researcher and visiting lecturer at the University of California, Berkeley. From 1994 to 1998, she worked for TCSI and ATMEL in Berkeley, CA. In 1998, she joined the faculty of University of California, Los Angeles (UCLA). She is currently a professor at the K.U.Leuven and an adjunct professor at UCLA. At K.U.Leuven, she is a codirector of the Computer Security and Industrial Cryptography (COSIC) Laboratory. Her research interests include circuits, processor architectures and design methodologies for real-time embedded systems for security, cryptography, digital signal processing, and wireless communications. This includes the influence of new technologies and new circuit solutions on the design of next-generation systems on chip. She was the program chair of the Ninth International Workshop on Cryptographic Hardware and Embedded Systems (CHES 07), the 19th IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP 08), and the ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED 02). She was also the general chair of ISLPED 2003. She was a member of the executive committee of the 42nd and 43rd Design Automation Conference (DAC) as the design community chair.