Robustness Analysis of Watermark Verification Techniques for FPGA Netlist Cores*

Daniel Ziener, Moritz Schmid, and Jürgen Teich

Hardware/Software Co-Design
Department of Computer Science
University of Erlangen-Nuremberg, Germany
Am Weichselgarten 3
91058 Erlangen, Germany
email: {*daniel.ziener, moritz.schmid, teich*}*@cs.fau.de*

Abstract. In this paper we analyze the robustness of watermarking techniques for FPGA IP cores against attacks. Unlike most existing watermarking techniques, the focus of our techniques lies on ease of verification, even if the protected cores are embedded into a product. Moreover, we have concentrated on higher abstraction levels for embedding the watermark, particularly at the logic level, where IP cores are distributed as netlist cores. With the presented watermarking methods, it is possible to watermark IP cores at the logic level and identify them with a high likelihood and in a reproducible way in a purchased product from a company that is suspected to have committed IP fraud. For robustness analysis we enhanced a theoretical watermarking model, originally introduced for multimedia watermarking. Finally, two exemplary watermarking techniques for netlist cores using different verification strategies are described and the robustness against attacks is analyzed.

1 Introduction

The ongoing miniaturization of on-chip structures allows us to implement very complex designs which require very careful engineering and an enormous effort for debugging and verification. Indeed, complexity has risen to such enormous measures that it is no longer possible to keep up with productivity constraints if all parts of a design must be developed from scratch. In addition, the very lively market for embedded systems with its demand for very short product cycles intensifies this problem significantly. A popular solution to close this so called productivity gap is to reuse design components that are available in-house or that have been acquired from other companies. The constantly growing demand for ready to use design components, also known as IP cores, has created a very lucrative and flourishing market which will continue its current path not only into the near future.

* Published in: "Design Methodologies for Secure Embedded Systems", LNEE 78, pp. 105-127, Springer-Verlag Berlin Heidelberg, 2010.

One problem of IP cores is the lack of protection mechanisms against unlicensed usage. A possible solution is to hide a unique signature (watermark) inside the core by which the original author can be identified and an unlicensed usage can be proven. Our vision is that it should be possible to detect the unlicensed usage of an IP core solely using the product in which the IP core may be embedded and the watermark information of the original author. It should not be necessary to request any additional information from the manufacturer of suspicious product. Such concepts of course need advanced verification techniques in order for a signature or certain characteristics to be detectable in one of possibly many IP cores inside a system. Another aspect to be considered is the fact that IP cores will undergo several sophisticated optimization steps during the course of synthesis. It is of utmost importance that a watermark is transparent towards design and synthesis tools, that is, the embedded identification must be preserved in all possible scenarios. Whilst on the one hand, we must deal with the problem that automated design tools might remove an embedded signature all by themselves, a totally different aspect is that embedded signatures must also be protected against the removal by illegitimate parties whose intention is to keep the IP core from being identifiable. The latter is not to be taken lightly because if a sufficiently funded company decides to use unlicensed cores to, for example, lower design costs, there are usually have very high skilled employees assigned with the task to remove or bypass the embedded watermark.

Figure 1 depicts a possible watermarking flow. An IP core developer embeds a signature inside his IP core using a watermark embedder and publishes the protected IP core. The intention of this procedure is that companies interested into using the developer's core would obtain a licensed copy. However, a third-party company may also obtain an unlicensed copy of the protected IP core and use it in one of their products. If the IP core developer becomes suspicious that his core might have been used in a certain product without proper licensing, he can simply acquire the product and check for the presence of his signature. If this attempt is successful and his signature presents a strong enough proof of authorship, the developer may decide to accuse the product manufacturer of IP fraud and press legal charges.

IP cores exist for all design flow levels, from plain text HDL cores on the register-transfer level (RTL) to bitfile cores for FPGAs or layout cores for ASIC designs on the device level. In the future, IP core companies will concentrate more and more on the versatile HDL and netlist cores due to their flexibility. One reason for this development is that these cores can be easily adapted to new technologies and different FPGA devices. This work focuses on watermarking methods for IP cores implemented for FPGAs. These have a huge market segment and the inhibition threshold for using unlicensed cores is lower than in the ASIC market where products are produced in high volumes and vast amounts of funds are spent for mask production. Moreover, we concentrate on flexible IP cores which are delivered on the logic level in a netlist format. The advantage of this form of distribution is that these cores can be used for different families FPGA devices and can be combined with other cores to obtain a complete SoC

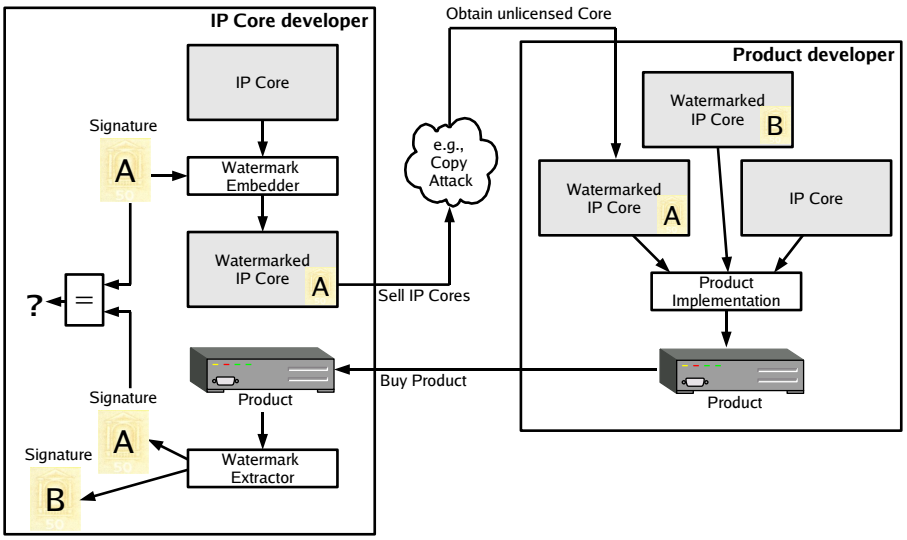

Fig. 1. An IP core developer embeds a watermark inside his core. If a company uses this core in one of their product without proper licensing, the IP core developer can obtain the product and check it for the presence of his watermark.

solution. Our work differs from most other existing watermarking techniques, which do not cover the area of HDL and netlist cores, or are not able to easily extract an embedded watermark from a heterogeneous SoC implemented in a given product.

The remaining work is organized as follows: In Section 2, a short overview of related work for IP watermarking is provided. Afterwards, Section 3 presents a theoretical model for watermarking IP cores. Section 4 deals with different strategies to extract a watermark from an FPGA embedded into a product. We proceed by describing two methods for extracting a watermark. The first method explains the extraction of a watermark from an FPGA bitfile in Section 5. Analyzing the power consumption of the FPGA in order to verify the presence of a watermark is the second method and will be discussed in Section 6. Additionally, the robustness against typical attacks will be analyzed for both methods. In conclusion, the contributions will be summarized.

2 Related Work

IP cores are often distributed like software and can therefore be used without proper legitimacy, which ranges from over-provisioning agreed on amounts of licensed uses to simply not licensing an IP core at all. Some core suppliers use encryption to protect their cores. The downside of such approaches is that encrypted cores can only be used in conjunction with special tools and that the

encryption will eventually be broken by individuals with high criminal energy. A different approach is to hide a signature in the core, a so-called watermark, which can be used as a reactive proof of the original ownership enabling IP core developers to identify and react upon IP fraud. There exist many concepts and approaches on the issue of integrating a watermark into a core, several of which will be reviewed in this section.

In general, hiding a signature into data, such as a multimedia file, some text, program code, or even an IP core by steganographic methods is called watermarking. For multimedia data, it is possible to exploit the imperfection of human eyes or ears to enforce variations on the data that represent a certain signature, but for which the difference between the original and the watermarked work cannot be recognized. Images, for example, can be watermarked by changing the least significant bit positions of the pixel tonal values to match the bit sequence of the original authors signature. For music, it is a common practice to watermark the data by altering certain frequencies, the ear cannot perceive and thus not interfering with the quality of the work [5]. In contrast, watermarking IP cores is entirely different from multimedia watermarking, because the user data, which represents the circuit, must not be altered since functional correctness must be preserved.

Most methods for watermarking IP cores focus on either introducing additional constraints on certain parts of the solution space of synthesis and optimization algorithms, or adding redundancies to the design.

Additive methods add a signature to the functional core, for example, by using unused lookup-tables in an FPGA [15, 19] or by sending the signature as a preamble of the output of the test mode [8]. Constraint-based methods were originally introduced by [9] and restrict the solution space of an optimization algorithm by setting additional constraints which are used to encode the signature. Methods for constraint-based watermarking in FPGAs exploit the scan-chain [13], preserve nets during logic synthesis [12], place constraints for CLBs in odd/even rows [10], alter the transistor width [4] or route constraints with unusual routing resources [10].

A common problem of many watermarking approaches is that for verification of the presence of the marks, the existence and the characteristic of a watermark must be disclosed, which enables possible attackers to remove the watermark. To overcome this obstacle, Adelsbach [2] and Li [16] have presented so-called zero-knowledge watermark schemes which enable the detection of the watermark without disclosing relevant information.

A survey and analysis of watermarking techniques in the context of IP cores is provided by Abdel-Hamid and others [1]. Further, we refer to our own survey of watermarking techniques for FPGA designs [25]. Moreover, a general survey of security topics for FPGAs is given by Drimer [7].

3 Theoretical Watermark Model for Robustness Analysis against Attacks

In this section, we propose a theoretical model for IP core watermarking. With this model, different threats and attack scenarios can be described and evaluated. In general, watermarking techniques must deal with an uncontrolled area, where the watermarked work is further processed. This is true for multimedia watermarking, where, for example, watermarked images are processed to enhance the image quality by filters or for IP core watermarking where the core is combined with other cores and traverses other design flow steps. However, the watermarked work may be exposed to further attacks in this uncontrolled area that may destroy the watermark and thus the proof of authorship as well. This uncontrolled area is difficult to describe in a precise way and therefore, the security goals and issues for watermarking are often given in natural language which results in an imprecise description. This natural description makes an assessment of the security very difficult, particularly if the attackers are intelligent and creative.

Introducing a defined theoretical watermarking model with attackers and threats allows us to assess the security of IP core watermarking techniques. However, it should be noted that the model has to cover all possible attack scenarios and represent all aspects of the real world behavior to allow for a meaningful assessment of the security. In this section, we present a general watermark model introduced by Li et. al. [17] which will be enhanced with aspects of IP core watermarking.

Watermarking intellectual property can be specified precisely by characterizing the involved actions using a security model. We use the standard definitions from security theory, which defines security goals, threats and attacks. Security goals represent certain abilities of a scheme, which are important to protect in order to keep its functionality in tact. These abilities may be violated by threats which are realized by attacks. Regarding watermarking, the overall security goal is to be able to present a proof of authorship that is strong enough to hold in front of a court. The security goal of a watermark scheme is violated if the original author cannot produce a strong enough proof of authorship, so that a dispute with another party will lead to an ownership deadlock, but also in the occasion, where another party is able to present a more convincing proof of authorship than the original author, resulting in counterfeit ownership. Another violation of the proof of authorship occurs if the watermark of a credible author is forged by another author and is used to convince a third party, that a work was created by someone who did not.

An attacker can realize an ownership deadlock, if he can present a watermark in the work, that is at least as convincing as the original authors watermark. If such an ambiguity attack is successful, the real ownership cannot be decided and the original author cannot prove his authorship. If, in addition, the ambiguity attack results in the pirate being able to present an even more convincing proof of authorship than the creator of the work, the pirate can counterfeit the ownership. Another way to take over the ownership of a piece of IP is to be able to remove the

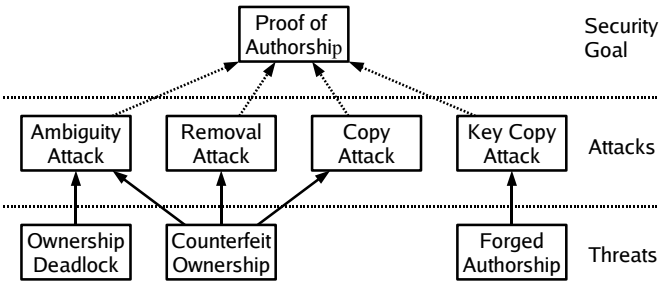

Fig. 2. An overview of threats, attacks and the watermarking security goal of the proof of authorship. The different threats are realized by attacks which violate the security goal.

original authors watermark by means of a removal attack. Forged authorship can be achieved by a key copy attack which simply duplicates the means of creating a credible authors watermark. One last violation of the security goal does not directly involve the author, but requires him to not take part in a dispute over theft. The theft of a work resulting in counterfeit ownership can be realized by a copy attack. The realized threat is only successful until the original author realizes the violation. An overview of the introduced terms can be observed in Figure 2.

Watermarking IP cores in electronic design automation is in some aspects different from multimedia watermarking (image, audio, etc.). An essential difference is that watermarking should preserve the functionality of the core. Another difference is that IP cores can be distributed at several abstraction levels which have completely different properties for the watermark security against attacks. We define different design steps as different technology or abstraction levels a work or IP core can be specified on. On higher abstraction levels, as for example on the architecture or register-transfer level, the functionality is described by an algorithm. At these levels, mainly the behavior is described and the representation is optimized for easy reading and understanding the algorithm. During the course of the design flow, more and more information is added. For example, placement information is included at the device level representation of the core. Extracting only the relevant information about the behavior of the algorithm is much harder than at higher abstraction levels. Furthermore, the information at lower abstraction levels is usually interpreted by tools rather than humans. The representation of this information is therefore optimized for machine and not human readability. For example, consider an FPGA design flow. Here, three different abstraction levels exist: RTL, logic, and device level. An algorithm, specified on the register-transfer-level (RTL) in an HDL core is easier to understand than a synthesized algorithm on the logic level, represented by a netlist. In

summary, we can say that the behavior of an algorithm is easier to understand on higher abstraction levels than it is on the lower ones.

Transformations from a higher to a lower abstraction level are usually done by design tools. For example, a synthesis tool is able to transform an HDL core specified on the register-transfer level (RTL) into its representation on the logic level. Transformations from a lower to a higher level can be achieved by reverse engineering. Here, usually no common tools are available. One exception is the Java library *JBits* from Xilinx [21] which is able to interpret the bitfiles of Virtex-II device types. Thus, it is possible to transfer a bitfile core into a netlist at the logic level by using JBits. However, in general, reverse engineering must be considered as very challenging task which may cause high costs.

A watermark can be embedded at every abstraction level. Furthermore, the watermarked core can be published and distributed also at every abstraction level which must not necessarily be the same level at which the watermark was embedded. However, the extraction of the watermark is usually done on the lowest abstraction level, because this is the representation of the design which is implemented into the end product.

Hiding a watermark at a lower abstraction level is easier, because first, there are more possibilities of how and where to hide the watermark and second, the information stored at these abstraction levels is usually outside the reception area of the human developer.

To explain all threats and attacks in detail, some definitions have to be made first [17, 20].

3.1 Definitions

Our definitions for the IP core watermarking model [20] are derived from the general watermarking model introduced by Li et al. [17]. A work or IP core that is specified at abstraction level Y is denoted by $I_Y = (x_{Y_1}, x_{Y_2}, \ldots, x_{Y_m})$, where each $x_{Y_i} \in \mathcal{I}_Y$ is an element of the work, and $\mathcal{I}_Y$ is a universe, inherent to the abstraction level Y. For example, an FPGA design at the device abstraction level might be represented by a bitfile which can be characterized as a work $I_B = (x_{B_1}, \ldots, x_{B_m})$, whose elements reside in the universe *Bit* ($\mathcal{I}_B$). Hence, a bitfile I_B with $|I_B| = m$ can also be considered as a binary sequence $I_B = \{0,1\}^m$.

Let $\mathcal{T}(\cdot)$ be a transformation, which transforms a work on a specific abstraction level into a work of another abstraction level. A transformation from the higher level Y to the lower abstraction level Z is denoted $\mathcal{T}_{Y \rightarrow Z}(\cdot)$, whereas a transformation from a lower to a higher level is denoted $\mathcal{T}_{Y \leftarrow Z}(\cdot)$.

Let $Dist_Y(\cdot,\ \cdot)$ be a distance function which is able to measure the differences of two works of the same abstraction level. If the distance of two IP cores I_Y and I'_Y of the same abstraction level Y is smaller than a threshold value t_I ($Dist_Y(I_Y, I'_Y) < t_I$), the two works may be considered similar.

A watermark W_Y is a vector $W_Y = (w_{Y1}, w_{Y2}, \ldots, w_{Yl})$, where each element $w_{Yi} \in \mathcal{W}_Y$. The universe $\mathcal{W}_Y$ is dependent on the universe of the work $\mathcal{I}_Y$ and the watermark generation process. A key K is a sequence of m binary bits ($K = \{0,\ 1\}^m$).

In the watermark model, there exist three algorithms: the *watermark generator* $\mathcal{G}$, the *watermark embedder* $\mathcal{E}$, and the *watermark detector* $\mathcal{D}$. In detail, a specific watermark generator $\mathcal{G}_X(\cdot)$ is able to generate a watermark W_X for the abstraction level X from a key K: $W_X = \mathcal{G}_X(K)$. The input of the watermark embedder or detector must be in the same abstraction level. For example, to watermark an IP core I_X at abstraction level X, also the watermark W_X must be generated for this abstraction level. So to obtain a watermarked work on the abstraction level X, it is necessary to also use a watermarked and an embedding algorithm suitable for the same abstraction level, i.e., $\widetilde{I}_X = \mathcal{E}_X(I_X, W_X)$. The watermark in $\widetilde{I}_X$ should obviously not be visible. Therefore, the difference between I_X and $\widetilde{I}_X$ should be small. With the distance function, this can be expressed as $Dist_X(I_X, \widetilde{I}_X) < t_I$, where t_I is a threshold value upon the difference is noticeable. Using the watermark detector $\mathcal{D}_X$, the existence of the watermark W_X in the work $\widetilde{I}_X$ can be proven, if $\mathcal{D}_X(\widetilde{I}_X, W_X) = true$ or negated if $\mathcal{D}_X(\widetilde{I}_X, W_X) = false$.

In order to achieve full transparency of the watermarking process towards design tools, it is an essential requirement that a work, marked on any abstraction level, will retain the watermark if transformed to a lower abstraction level. Hence, if $\mathcal{D}_Y(\widetilde{I}_Y, W_Y) = true$, so should also $\mathcal{D}(\widetilde{I}_Z, W_Z) = true$, if $\widetilde{I}_Z = \mathcal{T}_{Y \to Z}(\widetilde{I}_Y)$, and W_Z is a representation of W_Y on abstraction level Z.

However, considering reverse engineering, the watermark information may be removed by the reverse engineering transformation $\mathcal{T}_{Y \leftarrow Z}(\cdot)$, or the detection and removal of the watermark may be greatly simplified on the higher abstraction level. For example, consider an FPGA bitfile IP core watermarking technique for which the watermark is stored in some placement information inside the bitfile. The watermark is generated for bitfiles on the device level: $W_B = \mathcal{G}_B(K)$ and is embedded in a bitfile core I_B to create the watermarked bitfile: $\widetilde{I}_B = \mathcal{E}_B(I_B, W_B)$. If an attacker is able to reverse engineer the bitfile and reconstruct a netlist on the logic level, the placement information will get lost, since there is no representation for this information on the logic level. This implies, of course, that the watermark is lost, as well: $\widetilde{I}_L = \mathcal{T}_{L \leftarrow B}(\widetilde{I}_B)$, $\mathcal{D}_L(\widetilde{I}_L, W_L) = false$. Another problem of reverse engineering may be that an embedded watermark might become obviously readable at the higher abstraction level and can be removed easily.

Figure 3 shows an example of the IP core watermark model considering different abstraction levels.

3.2 Threat Model

In the general multimedia watermarking model introduced by Li et al. [17], it should be computationally infeasible to remove the watermark without changing the properties of the work. For the introduced IP core watermarking model, this requirement does not necessarily hold. Sometimes, it might be easier for an attacker to redevelop an IP core than to remove a watermark. The question to purchase or to redevelop a core is a pure matter of cost. An uprising economical

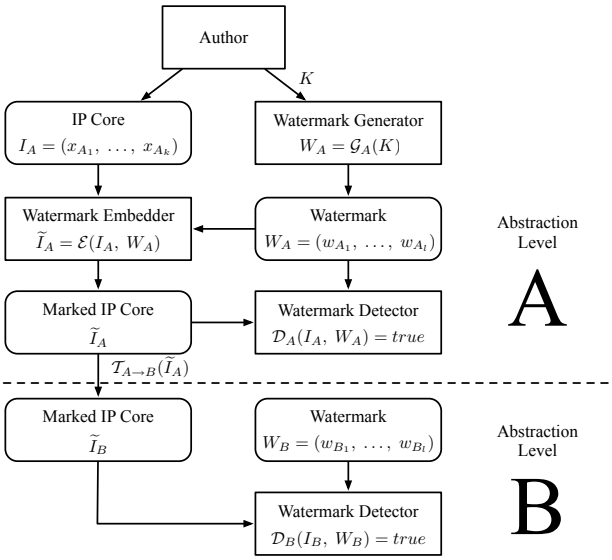

Fig. 3. An example of a watermarking procedure characterized in the IP core watermark model with different abstraction levels. At abstraction level A, the watermark is generated and embedded. A transformation to the abstraction level B retains the watermark [20].

question is whether the development of an attack is an option. For many cases, the redevelopment from scratch might be cheaper than obtaining an unlicensed core and develop an attack in order to remove the watermark. On the other hand, there are designs involving such cunning cleverness and creativity that trying to redevelop a work of equivalent economic value would exceed the costs of developing an appropriate attack by several orders of magnitude.

We may consider a watermarking technique secure, if the cost for obtaining an unlicensed IP core and developing a removal attack is higher than to purchase the IP core.

Let $\mathcal{A}_Y$ be an algorithm which is able to transform a watermarked IP core $\widetilde{I}_Y$ at the abstraction level Y into an IP core with removed or disabled watermark $I'_Y = \mathcal{A}_Y(\widetilde{I}_Y)$. Let $C(\cdot)$ be a cost function. Furthermore, denote $C_D(\cdot)$ as the development cost of a specified IP core or attack and $C_P(\cdot)$ the purchase cost of an IP core. Let $C_O(\cdot)$ denote the cost to obtain an (unlicensed) IP core. Note that this cost may vary between the costs for copying the core from an arbitrary source and those for purchasing it. We define a watermarked core $\widetilde{I}_Y$ to be secure against attacks if attacks produce higher costs than the legal use of the core. Instead of requiring computational infeasibility, it is enough to fulfill:

$$C_P(\widetilde{I}_Y) < C_D(I_Y) \leq (C_O(\widetilde{I}_Y) + C_D(\mathcal{A}_Y)). \tag{1}$$

Furthermore, a reverse engineering step to a higher abstraction level and the development of an attacker algorithm on this level might be cheaper than the

development of an attacker algorithm on the lower abstraction level. Therefore, we must also consider the usage of reverse engineering:

$$C_P(\widetilde{I}_Y) < C_D(I_Y) \leq (C_O(\widetilde{I}_Y) + C(\mathcal{T}_{X \leftarrow Y}(\widetilde{I}_Y)) + C_D(\mathcal{A}_X(\widetilde{I}_X)) + C(\mathcal{T}_{X \rightarrow Y}(I'_X)). \tag{2}$$

Definition 1. *An IP core watermarking scheme is called t_I-resistant to removal attacks if for any attacker $\mathcal{A}$ and any IP core $\widetilde{I}_Y$ of a given abstraction level Y and watermarked by W_Y, it is either computationally infeasible to compute $I'_Y = \mathcal{A}(\widetilde{I}_Y)$ with $Dist_Y(\widetilde{I}_Y, I'_Y) < t_I$ and $\mathcal{D}_Y(I'_Y, W_Y) = false$ or produces higher costs than its legal use.*

The term *t_I-resistant* means that the watermark scheme is resistant against removal attacks with respect to the threshold value t_I. If the distance exceeds t_I, the works cannot be counted as identical. For example, if the attacker creates a completely new work, the watermark is also removed, but the works are not the same. The phrase *computationally infeasible* follows the standard definition from cryptography. Something is computationally infeasible if the cost (e.g., memory, runtime, area) is finite but impossibly large [6]. Here, this is true if the probability $Pr[\mathcal{A}_Y(\widetilde{I}_Y) = I'_Y]$ is negligible with respect to the problem size n. A quantity X is negligible with respect to n if and only if for all sufficiently large n and any fixed polynomial $q(\cdot)$ (the attacker $\mathcal{A}_Y$ is defined as an algorithm of polynomial complexity), we have $X < 1/q(n)$ [17].

In other words, with a sufficiently large problem size of watermarked work $\widetilde{I}_Y$, resistance against removal attacks means that the attacker is unable to remove the watermark as the problem size is beyond the computational capability of the attacker, unless the resulting work is perceptually different from the original work.

For ambiguity attacks where an attacker tries to counterfeit the ownership or to achieve an ownership deadlock, the attacker searches for a fake watermark inside the IP core. This can be done by analyzing the IP core and searching for a structural or statistical feature which might be suitable to be interpreted as a fake watermark. However, the published IP core may be delivered in different target technology versions, for example, for an ASIC design flow or for different FPGA target devices. This fake watermark must of course be present in any other distributed version of the IP core in order to guarantee the attacker's authentic evidence of ownership. Furthermore, the attacker must present a fake original work and the evidence of a comprehensible watermark generation from a unique key, which clearly identifies the attacker. These are all reasons why ambiguity attacks are very difficult in the area of IP core watermarking.

Definition 2. *An IP core watermarking scheme is called resistant to ambiguity attacks if for any attacker $\mathcal{A}$ and any given IP core $\widetilde{I}_Y$ of a certain abstraction level Y and watermarked by W_Y, it is either computationally infeasible to compute a valid watermark W'_Y such that $\mathcal{D}_Y(\widetilde{I}_Y, W'_Y) = true$ or produces more costs than its legal use.*

In case of key copy attacks, the key of a credible author is used to watermark a work with lower quality. In general, it should be impossible for an attacker to create a work I'_Y which is distinguishable from any work of another author where the key or watermark of a credible author can be found.

Definition 3. *A watermarking scheme is t_I-resistant to key copy attacks if for any attacker $\mathcal{A}_Y$ and any work $\tilde{I}_Y = \mathcal{E}_Y(I_Y, W_Y)$ for some original I_Y and the watermark W_Y, it is computationally infeasible for $\mathcal{A}_Y$ to compute a work I'_Y such that $Dist(I_Y, I'_Y) > t_I$, yet $\mathcal{D}_Y(I'_Y, W_Y) = true$ [17].*

To prevent key copy attacks, a private/public key algorithm, like RSA [18] can be used. RSA is an asymmetrical cryptography method that is based on factorization of a number into prime numbers. The author encrypts a message which clearly identifies the author and the work with his private key. The work can be identified by a hash value over the original work. This encrypted message is now used for generating the watermark and embedded inside the work. Stealing this watermark is useless, because everyone can decrypt the message with the public key, whereas no one can alter this message.

4 Watermark Verification Strategies for Embedded FPGAs

The problem of applying watermarking techniques to FPGA designs is not the coding and insertion of a watermark, rather it is the verification with an FPGA embedded in a system that poses the real challenge. Hence, our methods concentrate in particular on the verification of watermarks. When considering finished products, there are five potential sources of information that can be used for extracting a watermark: The configuration bitfile, the ports, the power consumption, electromagnetic (EM) radiation, and the temperature.

If the developer of an FPGA design has disabled the possibility to simply read back the bitfile from the chip, it can be extracted by wire tapping the communication between the PROM and the FPGA. Some FPGA manufactures provide an option to encrypt the bitstream which will be decrypted only during configuration inside the FPGA. Monitoring the communication between PROM and FPGA in this case is useless, because only the encrypted file will be transmitted. Configuration bitfiles mostly use a proprietary format which is not documented by the FPGA manufacturers. However, it seems to be possible to read out some parts of the bitfile, such as information stored in RAMs or lookup tables. In Section 5, we introduce a procedure in which the watermarks are inserted into an IP core specified on the logic level in form of a netlist and can then be extracted from the configuration bitstream.

Another popular approach for retrieving a signature from an FPGA is to employ unused ports. Although this method is applicable to top-level designs, it is impractical for IP cores, since these are mostly used as components that will be combined with other resources and embedded into a design so that the

ports will not be directly accessible any more. Due to these restrictions, we do not discuss the extraction of watermarks over output ports.

Furthermore, it is possible to force patterns on the power consumption of an FPGA, which can be used as a covert channel to transmit data to the outside of the FPGA. We have shown in [27] and [24] that the clock frequency and toggling logic can be used to control such a power spectrum covert channel. The basic idea to use these techniques for watermarking is to force a signature dependent toggle pattern and extract the resulting change in power consumption as a signature from the FPGA's power spectrum. We refer to this method as "Power Watermarking" in Section 6

With almost the same strategy it is also possible to extract signatures from the electro magnetic (EM) radiation of an FPGA. A further advantage of this technique is that a raster scan of an FPGA surface with an EM sensor can also use the location information to extract and verify the watermark. Unfortunately, more and more FPGAs are delivered in a metal chip package which absorbs the EM radiation. Nevertheless, this is an interesting alternative technique for extracting watermarks and invites for future research.

Finally, a watermark might be read out by monitoring the temperature radiation. The concept is similar to the power and EM-field watermarking approaches, however, the transmission speed is drastically reduced. Interestingly, this is the only watermarking approach which is commercially available [11]. Here, reading the watermark from an FPGA may take up to 10 minutes.

5 Watermark Verification using the FPGA Bitfile

In this section we present a method where an embedded signature is extracted from an FPGA bitfile. We start out by discussing how the contents of the lookup tables may be extracted from the FPGA bitfile. Following, a watermarking method for netlist cores is proposed (see also [20]).

5.1 Lookup Table Content Extraction

In order to harden the watermark against removal it is very important to integrate the watermark into the functional parts of the IP core, so that simply removing the mark carrying components would damage the core. For FPGA designs, the functional lookup tables are an ideally suited component for carrying watermarks. From a finished product, it is possible to obtain the configuration bitstream of the FPGA. The extraction of the lookup table contents from the configuration bitfile depends on the FPGA device and the FPGA vendor. To read out the LUT content directly from the bitfile, it must be known at which position in the bitfile the lookup table content is stored and how these values must be interpreted. In [22], for example, a standard black-box reverse engineering procedure is applied to interpret Xilinx Virtex-II and Virtex-II Pro bitfiles. To generalize this approach, we define a lookup table extractor function $\mathcal{L}_X(\cdot)$ for the abstraction level X. The extractor function is able to extract the lookup

table content of a work I_X as follows: $\mathcal{L}_X(I_X) = \{x_{X_1}, x_{X_2}, \ldots, x_{X_m}\}$, whereas x_{X_i} is a lookup table content element of the abstraction level X, and m is the number of used lookup tables. The extraction function can be applied to extract the lookup table contents of a design I_B of the bitfile on abstraction level B: $\mathcal{L}_B(I_B) = \{x_{B_1}, x_{B_2}, \ldots, x_{B_q}\}$. Each element x_{B_i} consists of the lookup table content as well as the (x_S, y_S) coordinates of the corresponding lookup table.

5.2 Watermarks in Functional LUTs for Netlist Cores

Since we want to keep the IP core as versatile as possible, we watermark the design in the form of a netlist representation, which, although technology dependent to a certain degree, can still be used for a large number of different devices. Netlist designs will almost certainly undergo the typical design flow for silicon implementations. This also includes very sophisticated optimization algorithms, which will eliminate any redundancy that can be found in the design in order to make improvements. As a consequence it is necessary to embed the watermarks in the netlist in such a way, that the optimization tools will not remove the watermarks from the design. In Xilinx FPGAs, for example, lookup tables are essentially RAM cells, with the inputs specifying which of the stored bits to deliver to the output of the RAM. Naturally, these cells can therefore also be used as storage, but also as shift-register cells (see Figure 4). Interesting, however, is the fact that if the cell is configured as a lookup table, Xilinx optimization tools will try to optimize the contained logic function. If the cell is in contrast configured as a shift-register or distributed RAM, the optimization tools will leave the contents alone, but the logic function is still carried out. This means, that if we want to add redundancy to a netlist, that is not removed by automized tools, all we have to do is to take the corresponding cells out of the scope of the tools. FPGAs usually consist of the same type of lookup tables with respect to the number of inputs. For example, the Xilinx Virtex-II uses lookup tables with four inputs whereas the Virtex-5 has lookup tables with six inputs. However, in common netlist cores many logical lookup tables exist, which have less inputs than the type used on the FPGA. These lookup tables are mapped to the physical lookup tables of the FPGA during synthesis. If the logical lookup table of the netlist core has fewer inputs than the physical representation, the memory space which was not present in the logical representation remains unused. Using the unused memory space of functional lookup tables for watermarking without converting the lookup table either to a shift register or distributed memory turns out to be not applicable, because design flow tools identify the watermark as redundant and remove the content due to optimization. Converting the watermarked functional lookup table into a shift register or a memory cell prevents the watermark from deletion due to optimization.

Embedding the Watermark The first step of embedding a watermark is to extract all lookup tables from a given netlist core I_L: $\mathcal{L}_L(I_L) = \{lut_{L_1}, lut_{L_2}, \ldots, lut_{L_r}\}$, where L denotes the logic abstraction level used for netlist cores (see Figure 5). Each element lut_{L_i} denotes a lookup table primitive cell in the netlist

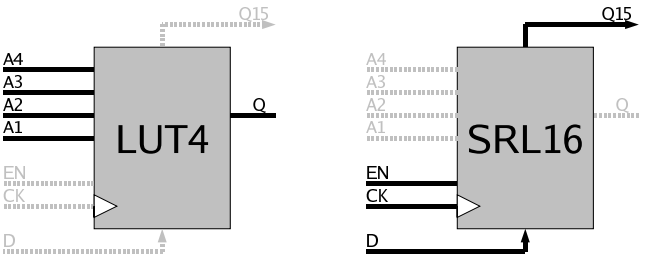

Fig. 4. In the Xilinx Virtex architecture, a lookup table (LUT4) can also be configured as a 16-bit shift-register lookup table(SRL16).

(e.g. for Virtex-II devices, LUT1, LUT2, LUT3, or LUT4). A watermark generator $\mathcal{G}_L(\cdot,\cdot)$ must know the different lookup table cells with the functional content as well as the unique key K to generate the watermarks: $\mathcal{G}_L(K, \mathcal{L}_L(I_L)) = W_L$.

From the unique key K a secure pseudo random sequence is generated. Some or all of the extracted lookup table primitive cells are chosen to carry a watermark. Note that only lookup tables from the netlist core can be chosen which use less inputs than the physical lookup tables on the FPGA. Usually a core which is worth to be watermarked consists of many markable lookup tables. Now, the lookup tables are transformed to shift registers, ordered, and the first 4 bits of the free space are used for a counter value. The other bits are initialized according to the position with values from the pseudo random stream, generated from the key K. Note that the number of bits which can be used for the random stream depends on the original functional lookup table type.

The generated watermark W_L consists of the transformed shift registers: $W_L = \{srl_{L_1}, srl_{L_2}, \ldots, srl_{L_k}\}$ with $k \leq r$. The watermark embedder $\mathcal{E}_L$ inserts the watermarks into the netlist core I_L by replacing the corresponding original functional lookup tables with the shift registers: $\mathcal{E}_L(I_L, W_L) = \widetilde{I}_L$. The watermarked work $\widetilde{I}_L$ can now be published and sold.

Extraction of the Watermark The purchased core $\widetilde{I}_L$ can now be combined by a product developer with other purchased or self developed cores and implemented into an FPGA bitfile: $\widehat{I}_B = \mathcal{T}_{L\to B}(\widetilde{I}_L \circ I'_{L_1} \circ I'_{L_2} \circ \ldots)$ (see Figure 5). An FPGA which is programmed with this bitfile $\widehat{I}_B$ may be part of a product. If the product developer is accused of using an unlicensed core, the product can be purchased and the bitfile can be read out, e.g., by wire tapping. The lookup table content and the content of the shift registers can be extracted from the bitfile: $\mathcal{L}_B(\widehat{I}_B) = \{\widehat{x}_{B_1}, \widehat{x}_{B_2}, \ldots, \widehat{x}_{B_q}\}$.

The lookup table or shift register elements x_{B_i} belong to the device abstraction level B. The representation can differ from the representation of the same content in the logic abstraction level L. For example, in Xilinx Virtex-II FPGAs the encoding of the shift register differs from the encoding of lookup tables.

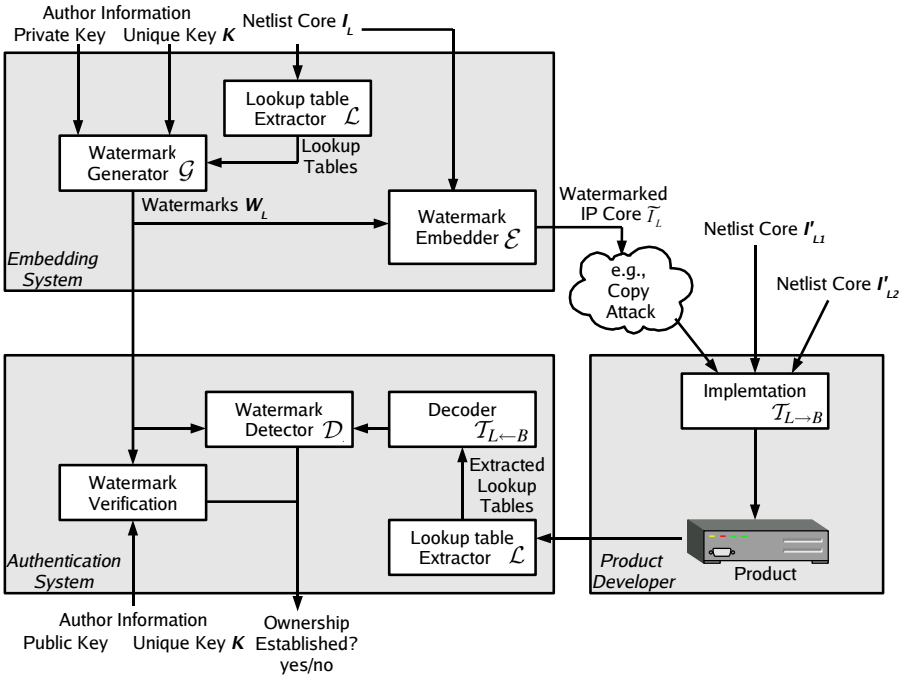

Fig. 5. The netlist core watermarking system. The embedding system is responsible for extracting the lookup tables from the netlist core, selecting suitable locations and embedding the watermarks in those LUTs that were converted into shift-registers. A product developer may obtain such a watermarked netlist core and combine it with other cores into a product. The lookup tables from the product can be extracted and transformed so that the detector can decide if the watermark is present or not.

For shift registers the bit order is reversed compared to the lookup table encodings. Therefore, the bitfile elements must be transferred to the logic level by the corresponding decoding. This can be done by the *reverse engineering operator*: $\mathcal{T}_{L\leftarrow B}(\mathcal{L}_B(\widetilde{I}_B)) = \{\widehat{x}_{L_1}, \widehat{x}_{L_2}, \ldots, \widehat{x}_{L_q}\}$. Reverse engineering lookup table or shift register content is however very simple compared to reverse engineering the whole bitfile. Now, the lookup table or shift register content can be used for the watermark detector $\mathcal{D}_L$ which can decide if the watermark W_L is embedded in the work or not: $\mathcal{D}_L(W_L, \{\widehat{x}_{L_1}, \widehat{x}_{L_2}, \ldots, \widehat{x}_{L_q}\}) = true/false$.

Robustness Analysis To recall Section 3, the most important attacks are *removal, ambiguity, key copy*, and *copy attacks*. As stated before, a possible protection against copy attacks does not exists and key copy attacks can be prevented by using an asymmetric cryptographic method, like RSA.

Removal attacks most likely occur on the logic level, after obtaining the unlicensed core and before the integration with other cores. The first step of a

removal attack is the detection of the watermarks. The appearance of the shift register primitive cells (here SRL16) in a netlist core is not suspicious because shift registers appear also in unwatermarked cores. However, the cumulative appearance may be suspicious, which may alert an attacker. In contrast to bitfiles, the signal nets can be easily read out from a netlist core. An attacker may analyze the net structures of shift registers in order to detect the watermarked cells. This might be successful, however, we can better hide the watermark if we alter the encoding of the watermark and, therefore, the connections to the watermark cell. The reachable functional part of the shift register can be shifted to other positions by using other functional inputs and clamping the remaining inputs to different values. If a watermark cell is detected by an attacker, he cannot easily remove the cell, because the cell also has a functional part. By removing the cell, the functional part is removed and the core is damaged. Therefore, after the detection of the watermark, the attacker must either decode the content of the watermarked shift register to extract the functional part and insert a new lookup table, or overwrite the watermarked part of the cell with other values, so the watermark is not detectable any more. The different encodings of the functional part of the shift register content complicates the analysis and the extraction of it. Furthermore, even if some watermarks are removed, the establishment of the right ownership of the core is still possible, because we need not all watermarked cells for a successful detection of the signature.

In case of *ambiguity attacks*, an attacker analyzes the bitfile or the netlist to find shift register or lookup table contents which may be suitable to build a fake watermark. However, the attacker must also present the insertion procedure to achieve a meaningful result. Due to the usage of secure one way cryptographic functions for generating the watermark, the probability of a success is very low. Furthermore, the attacker can use a self-written netlist core which he watermarked with his signatures and combine it with the obtained unlicensed core. The result is, that the watermarks of the authors of both cores are found in the bitfile, which are both trustful. Inside the unique key K, not only the author information should be included but also information of the core, e.g., a hash value over the netlist core file without the watermark. Of course, the attacker can use the identification of the obtained unlicensed core for watermarking his core. However, to generate a hash value of the obtained core without watermarks, he must first remove the marks. In general, attacks against this approach are possible, but they need a high amount of effort. To increase the security against ambiguity attacks, the core may be registered at a trusted third party.

6 Power Watermarking

This section describes watermarking techniques introduced in [27] and [24], where a signature is verified over the *power consumption pattern* of an FPGA. The presented idea is new and differs from [14] and [3] where the goal of using power analysis techniques is the detection of cryptographic keys and other security issues. For power watermarking methods, the term *signature* refers to the

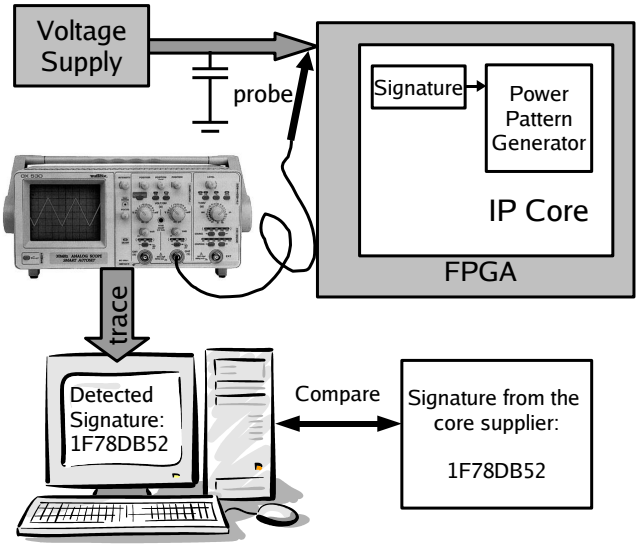

Fig. 6. Watermark verification using power signature analysis: From a signature (watermark), a power pattern inside the core will be generated that can be probed at the voltage supply pins of the FPGA. From the trace, a detection algorithm verifies the existence of the watermark.

part of the watermark which can be extracted and is needed for the detection and verification of the watermark. The signature is usually a bit sequence which is derived from the unique key for author and core identification.

There is no way to measure the relative power consumption of an FPGA directly. Only by measuring the relative supply voltage or current the actual power consumtion can be inferred. We have decided to measure the voltage of the core as close as possible to the voltage supply pins such that the smoothing from the plane and block capacities are minimal and no shunt is required. Most FPGAs have *ball grid array* (BGA) packages and the majority of them have vias to the back of the PCB for the supply voltage pins. So, the voltage can be measured on the rear side of the PCB using an oscilloscope. The voltage can be sampled using a standard oscilloscope, and analyzed and decoded using a program developed to run on a PC. The decoded signature can be compared with the original signature and thus, the watermark can be verified. This method has the advantage of being non-destructive and requires no further information or aids than the given product (see Figure 6).

In the power watermarking approach described in [26] and [27], the amplitude of the interferences in the core voltage is altered. The basic idea is to add a power pattern generator (e.g., a set of shift registers) and clock it either with

the operational clock or an integer division thereof. This power pattern generator is controlled according to the encoding of the signature sequence which should be sent.

The mapping of a signature sequence $s = \{0,1\}^n$ onto a sequence of symbols $\{\sigma_0, \sigma_1\}^n$ [24] is called encoding: $\{0,1\}^n \rightarrow \mathcal{Z}^n, n \geq 0$ with the alphabet $\mathcal{Z} = \{\sigma_0, \sigma_1\}$. Here, each signature bit $\{0,1\}$ is assigned to a symbol. Each symbol σ_i is a triple $(e_i, \delta_i, \omega_i)$, with the *event* $e_i \in \{\gamma, \bar{\gamma}\}$, the *period length* $\delta_i > 0$, and the *number of repetitions* $\omega_i > 0$. The event γ is *power consumption through a shift operation* and the inverse event $\bar{\gamma}$ is *no power consumption*. The period length is given in terms of number of clock cycles. For example, the encoding through 32 shifts with the period length 1 (one shift operation per cycle) if the data bit '1' should be sent, and 32 cycles without a shift operation for the data bit '0' is defined by the alphabet $\mathcal{Z} = \{(\gamma, 1, 32), (\bar{\gamma}, 1, 32)\}$.

Different power watermarking encoding schemes were introduced and analyzed in [27] and [24]. This includes the basic method with encoding scheme: $\mathcal{Z} = \{(\gamma, 1, 1), (\bar{\gamma}, 1, 1)\}$, the enhanced robustness encoding: $\mathcal{Z} = \{(\gamma, 1, 32), (\bar{\gamma}, 1, 32)\}$, the BPSK approach: $\mathcal{Z} = \{(\gamma, 1, \omega), (\bar{\gamma}, 1, \omega)\}$, and the correlation method with encoding $\mathcal{Z} = \{(\gamma, 25, 1), (\bar{\gamma}, 25, 1)\}$. To avoid interference from the operational logic in the measured voltage, the signature is only generated during the reset phase of the core.

The power pattern generator consists of several shift registers, causing a recognizable signature- and encoding-dependent power consumption pattern. As mentioned before in Section 5.2, a shift register can also be used as a lookup table and vice versa in many FPGA architectures (see Figure 4 in Section 5.2). A conversion of functional lookup tables into shift registers does not affect the functionality if the new inputs are set correctly. This allows us to use functional logic for implementing the power pattern generator. The core operates in two modes, the *functional mode* and the *reset mode*. In the functional mode, the shift is disabled and the shift register operates as a normal lookup table. In the reset mode, the content is shifted according to the signature bits and consumes power which can be measured outside of the FPGA. To prevent the loss of the content of the lookup table, the output of the shift register is fed back to the input, such that the content is shifted circularly. When the core changes to the functional mode, the content have to be shifted to the proper position to get a functional lookup table for the core.

To increase the robustness against removal and ambiguity attacks, the content of the power consumption shift register which is also part of the functional logic can be initialized shifted. Only during the reset state, when the signature is transmitted, the content of the functional lookup table can be positioned correctly. So, normal core operation cannot start before the signature was transmitted completely. The advantage is that the core is only able to work after sending the signature. Furthermore, to avoid a too short reset time in which the watermark cannot be detected exactly, the right functionality will only be established if the reset state is longer than a predefined time. This prevents the

user from leaving out or shorten the reset state with the result that the signature cannot be detected properly.

The signature itself can be implemented as a part of the functional logic in the same way. Some lookup tables are connected together and the content, the function of the LUTs, represents the signature. Furthermore, techniques described in Section 5.2 can be used to combine an additional watermark and the functional part in a single lookup table if not all lookup table inputs are used for the function. For example, LUT2 primitives in Xilinx Virtex-II devices can be used to carry an additional 12-bit watermark by restricting the reachability of the functional lookup table through clamping certain signals to constant values. Therefore, the final sending sequence consists of the functional part and the additional watermark. This principle makes it almost impossible for an attacker to change the content of the signature shift register. Altering the signature would also affect the functional core and thus result in a corrupt core.

The advantages of using the functional logic of the core as a shift register are the reduced resource overhead for watermarking and the robustness of this method. It is hard, if not impossible, to remove shift registers without destroying the functional core, because they are embedded in the functional design. The watermark embedder $\mathcal{E}_L(I_L, W_L) = \widetilde{I}_L$ consists of two steps. First, the core I_L must be embedded in a wrapper which contains the control logic for emitting the signature. This step is done at the register-transfer level before synthesis. The second step is at the logic level after the synthesis. A program converts suitable lookup tables (for example LUT4 for Virtex-II FPGAs) into shift registers for the generation of the power pattern and attaches the corresponding control signal from the control logic in the wrapper. The wrapper contains the control logic for emitting the watermark and a register that contains the signature. The ports of the wrapper are identical to the core, so we can easily integrate this wrapper into the hierarchy. The control logic enables the signature register while the core is in reset state. Also, the power pattern shift registers are shifted in correspondence to the current signature bit. If the reset input of the wrapper is deasserted, the core function cannot start immediately, but only as soon as the content in the shift registers has been shifted back to the correct position. Then the control logic deasserts the internal reset signal to enter normal function mode. The translation of four input lookup tables (LUT4) of the functional logic into 16 Bit shift registers (SRL16) is done at the netlist level. The watermarked core $\widetilde{I}_L$ is now ready for purchase or publication. A company may obtain an unlicensed version of the core $\widehat{I}_L$ and embeds this core in a product: $\widehat{I}_P = \mathcal{T}_{L \to B}(\widehat{I}_L \circ I'_{L_1} \circ I'_{L_2} \circ \ldots)$. If the core developer has a suspicious fact, he can buy the product and verify that his signature is inside the core using a detection function $\mathcal{D}_P(\widehat{I}_P, W_L) = true/false$. The detecting function depends on the encoding scheme. In [27] and [24], the detecting functions of all introduced encoding schemes are described in detail.

The advantage of power watermarking is that the signature can easily be read out from a given device. Only the core voltage of the FPGA must be measured and recorded. No bitfile is required which needs to be reverse-engineered.

Also, these methods work for encrypted bitfiles where methods extracting the signature from the bitfile fail. Moreover, we are able to sign netlist cores, because our watermarking algorithm does not need any placement information. However, many watermarked netlist cores can be integrated into on design. The results are superpositions and interferences which complicate or even prohibit the correct decoding of the signatures. To achieve the correct decoding of all signatures, we proposed *multiplexing* methods in [23].

Robustness Analysis The most common attacks against watermarking mentioned in Section 3 are *removal, ambiguity, key copy*, and *copy attacks*. Once again, key copy attacks can be prevented by asymmetric cryptographic methods, and there is no protection against copy attacks.

Removal attacks most likely take place on the logic level instead of the device level where it is really hard to alter the design. The signature and power shift registers as well as the watermark sending control logic in the wrapper are mixed with functional elements in the netlist. Therefore, they are not easy to detect. Even if an attacker is able to identify the sending logic, a deactivation is useless if the content of the power shift register is only shifted into correct positions after sending the signature. By preventing the sending of the watermark, the core is unable to start. Another possibility is to alter the signature inside the shift register. The attacker may analyze the netlist to find the place were the signature is stored. This attack is only successful if there is no functional logic part mixed with the signature. By mixing the random bits with functional bits, it is hard to alter the signature without destroying the correct functionality of the core. Therefore, this watermark technique can be considered as resistant against removal attacks.

In case of *ambiguity attacks*, an attacker analyses the power consumption of the FPGA in order to find a fake watermark, or to implement a core whose power pattern disturbs the detection of the watermark. In order to trustfully fake watermarks inside the power consumption signal, the attacker must present the insertion and sending procedure which should be impossible without using an additional core. Another possibility for the attacker is to implement a *disturbance core* which needs a lot of power and makes the detection of the watermark impossible. In [27] and [24], *enhanced robustness encoding methods* are presented which increase the possibility to decode the signature, even if other cores are operating during the sending of the signature. Although a disturbance core might be successful, this core needs area and most notably power which increases the costs for the product. The presence of a disturbance core in a product is also suspicious and might lead to further investigation if a copyright infringement has occurred. Finally, the attacker may watermark another core with his watermark and claim that all cores belong to him. This can be prevented by adding a hash value of the original core without the watermark to the signature like in the bitfile watermarking method for netlist cores. The sending of watermarks of multiple cores at the same time is addressed in [23].

7 Conclusions

In this paper, we have presented exemplary two different approaches for watermarking of IP cores. Our methods follow the strategy of an easy verification of the watermark or the identification of the core in a bought product from an accused company without any further information. Netlist cores, which have a high trade potential for embedded systems developers, are in the focus of our analysis. To establish the authorship in a bought product by watermarking or core identification, we have discovered different new techniques, how information can be transmitted from the embedded core to the outer world. In this paper, we concatenated on methods using the *FPGA bitfile* which can be extracted from the product and on methods where the signature is transmitted over the *power pins* of the FPGA. In Section 3, we adapt the *theoretical general watermark approach* from Li et al. [17] for IP core watermarking and show possible threats and attacks. Section 5 deals with IP core watermarking methods where the authorship is established by analysis of the extracted bitfile. In Section 6, we have described watermark techniques for IP cores where the signature can be extracted easily over the power pins of the chip. The main idea is that during a reset phase of a chip, a watermark circuit is responsible to emit a characteristic power pattern sequence that may be measured by voltage fluctuations on power pins. With these techniques, it is possible to decide with high confidence, whether an IP core of a certain vendor is present on the FPGA or not. For all methods, we analyzed the strengths and weaknesses in case of removal of ambiguity attacks.

References

1. Amr T. Abdel-Hamid, Sofiéne Tahar, and El Mostapha Aboulhamid. A Survey on IP Watermarking Techniques. *Design Automation for Embedded Systems*, 9(3):211–227, 2004.
2. André Adelsbach, Markus Rohe, and Ahmad-Reza Sadeghi. Overcoming the Obstacles of Zero-knowledge Watermark Detection. In *MM&Sec '04: Proceedings of the 2004 workshop on Multimedia and security*, pages 46–55, New York, NY, USA, 2004. ACM.
3. Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. The EM Side-Channel(s). In *CHES '02: 4th International Workshop on Cryptographic Hardware and Embedded Systems*, pages 29–45, London, UK, 2003. Springer-Verlag.
4. Fujun Bai, Zhiqiang Gao, Yi Xu, and Xueyu Cai. A Watermarking Technique for Hard IP Protection in Full-custom IC Design. In *International Conference on Communications, Circuits and Systems (ICCCAS 2007)*, pages 1177–1180, 2007.
5. Laurence Boney, Ahmed H. Tewfik, and Khaled N. Hamdy. Digital Watermarks for Audio Signals. In *International Conference on Multimedia Computing and Systems*, pages 473–480, 1996.
6. Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
7. Saar Drimer. Security for Volatile FPGAs. November 2009.

8. Y. C. Fan and H. W. Tsao. Watermarking for Intellectual Property Protection. *Electronics Letters*, 39(18):1316–1318, 2003.
9. Andrew Byun Kahng, John Lach, William Henry Mangione-Smith, Stefanus Mantik, Igor Leonidovich Markov, Miodrag M. Potkonjak, Paul Askeland Tucker, Huijuan Wang, and Gregory Wolfe. Constraint-Based Watermarking Techniques for Design IP Protection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(10):1236–1252, 2001.
10. Andrew Byun Kahng, Stefanus Mantik, Igor Leonidovich Markov, Miodrag M. Potkonjak, Paul Askeland Tucker, Huijuan Wang, and Gregory Wolfe. Robust IP Watermarking Methodologies for Physical Design. In *DAC '98: Proceedings of the 35th annual Design Automation Conference*, pages 782–787, New York, NY, USA, 1998. ACM.
11. Tom Kean, David McLaren, and Carol Marsh. Verifying the Authenticity of Chip Designs with the DesignTag System. In *HOST '08: Proceedings of the 2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, pages 59–64, Washington, DC, USA, 2008. IEEE Computer Society.
12. Darko Kirovski, Yean-Yow Hwang, Miodrag Potkonjak, and Jason Cong. Intellectual Property Protection by Watermarking Combinational Logic Synthesis Solutions. In *ICCAD '98: Proceedings of the 1998 IEEE/ACM international conference on Computer-aided design*, pages 194–198, New York, NY, USA, 1998. ACM.
13. Darko Kirovski and Miodrag Potkonjak. Intellectual Property Protection Using Watermarking Partial Scan Chains For Sequential Logic Test Generation. In *ICCAD '98: Proceedings of the 1998 IEEE/ACM international conference on Computer-aided design*, 1998.
14. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, pages 388–397, London, UK, 1999. Springer-Verlag.
15. John Lach, William H. Mangione-Smith, and Miodrag Potkonjak. Signature Hiding Techniques for FPGA Intellectual Property Protection. In *ICCAD '98: Proceedings of the 1998 IEEE/ACM international conference on Computer-aided design*, pages 186–189, New York, NY, USA, 1998. ACM.
16. Qiming Li and Ee-Chien Chang. Zero-knowledge Watermark Detection Resistant to Ambiguity Attacks. In *MMSec '06: Proceedings of the 8th workshop on Multimedia and security*, pages 158–163, New York, NY, USA, 2006. ACM.
17. Qiming Li, Nasir Memon, and Husrev T. Sencar. Security Issues in Watermarking Applications – A Deeper Look. In *MCPS '06: Proceedings of the 4th ACM international workshop on Contents protection and security*, pages 23–28, New York, NY, USA, 2006. ACM.
18. Ronald Linn Rivest, Adi Shamir, and Leonard Max Adleman. A Method for Obtaining Digital Signatures and Public-key Cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
19. Debasri Saha and Susmita Sur-Kolay. Fast Robust Intellectual Property Protection for VLSI Physical Design. In *ICIT '07: Proceedings of the 10th International Conference on Information Technology*, pages 1–6, Washington, DC, USA, 2007. IEEE Computer Society.
20. Moritz Schmid, Daniel Ziener, and Jürgen Teich. Netlist-Level IP Protection by Watermarking for LUT-Based FPGAs. In *Proceedings of IEEE International Conference on Field-Programmable Technology (FPT 2008)*, pages 209–216, Taipei, Taiwan, December 2008.
21. Xilinx Inc. JBits 3.0 SDK for Virtex-II. *URL: www.xilinx.com/labs/projects/jbits/*.

22. Daniel Ziener, Stefan Aßmus, and Jürgen Teich. Identifying FPGA IP-Cores based on Lookup Table Content Analysis. In *Proceedings of 16th International Conference on Field Programmable Logic and Applications (FPL 2006)*, pages 481–486, Madrid, Spain, August 2006.
23. Daniel Ziener, Florian Baueregger, and Jürgen Teich. Multiplexing Methods for Power Watermarking. In *Proceedings of the IEEE Int. Symposium on Hardware-Oriented Security and Trust (HOST 2010), Anaheim, USA*, June 2010.
24. Daniel Ziener, Florian Baueregger, and Jürgen Teich. Using the Power Side Channel of FPGAs for Communication. In *Proceedings of the 18th Annual International IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 2010)*, pages 237–244, May 2010.
25. Daniel Ziener and Jürgen Teich. Evaluation of Watermarking Methods for FPGA-Based IP-cores. Technical Report 01-2005, University of Erlangen-Nuremberg, Department of CS 12, Hardware-Software-Co-Design, Am Weichselgarten 3, D-91058 Erlangen, Germany, March 2005.
26. Daniel Ziener and Jürgen Teich. FPGA Core Watermarking Based on Power Signature Analysis. In *Proceedings of IEEE International Conference on Field-Programmable Technology (FPT 2006)*, pages 205–212, Bangkok, Thailand, December 2006.
27. Daniel Ziener and Jürgen Teich. Power Signature Watermarking of IP Cores for FPGAs. *Journal of Signal Processing Systems*, 51(1):123–136, April 2008.