

# New Directions for FPGA IP Core Watermarking and Identification

Daniel Ziener, Jürgen Teich

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

**Abstract.** In this paper, we present an overview of new watermarking and identification techniques for FPGA IP cores. 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. The investigated techniques establish the authorship by verification of either an FPGA bitfile or the power consumption of a given FPGA.

## 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 demands if all parts of a design must be developed from scratch. 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 is very likely to continue its current path not only into the near future.

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. However, there also exist techniques where an IP core can be identified without an additional signature. *Identification* methods are based on the extraction of unique characteristics of the IP core, e.g., lookup table contents for FPGA IP cores. With these techniques, the author of the core can be

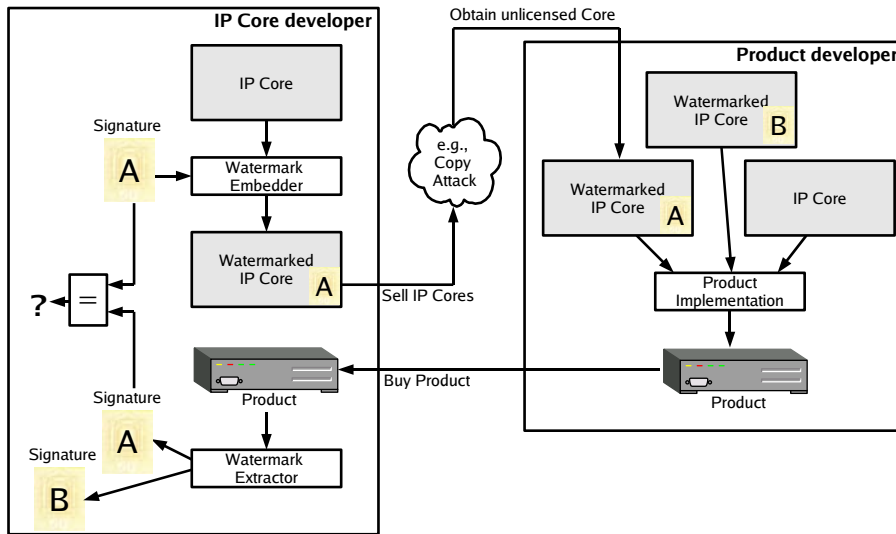
identified and an unlicensed usage can be proven. In this paper, watermarking as well as identification techniques for IP cores will be presented.

Our vision is that unlicensed IP cores, embedded in a complete SoC design which could be further embedded into a product, can be detected solely by using the given product and information from the IP core developer. Information of the accused SoC developer or product manufacturer should not be necessary and no extra information should be required from the accused company. Furthermore, the embedded author identification should be preserved even when the IP cores pass through different design flow steps. On the one hand, we must deal with the problem that design tools might remove the signature or the characteristics during synthesis and optimization. On the other hand, we must also secure the signature or characteristics against the removal by pirates which do not want the IP core be identifiable.

In Figure 1, a possible watermarking flow is depicted. An IP core developer embeds a signature inside his core using a *watermark embedder* and sells the protected IP core. A third-party company may 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 original core 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. 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 solution. Our methods differ from most other existing watermarking techniques, which do not cover the area of 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. Section 3 deals with different strategies to extract a watermark from an FPGA embedded into a product. We proceed by describing two ways for extracting a watermark. The first way explains the identification of an IP core from an FPGA bitfile in Section 4. 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 5. In conclusion, the contributions will be summarized.



**Fig. 1.** This figure shows a typical watermarking flow: An IP core developer embeds a watermark A inside his core. If a product developer obtains an unlicensed core and embeds this core in his product, the IP core developer can buy this product and extract the watermarks of all used IP cores. Now, he is able to compare his signature with the extracted signatures.

## 2 Related Work

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 [1]. 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 empty lookup-tables in an FPGA [2,3] or by sending the signature as a preamble of the output of the test mode [4]. Constraint-based methods were

originally introduced by [5] 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 [6], preserve nets during logic synthesis [7], place constraints for CLBs in odd/even rows [8], alter the transistor width [9] or route constraints with unusual routing resources [8].

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 [10] and Li [11] 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 [12]. Further, we refer to our own survey of watermarking techniques for FPGA designs [13]. Moreover, a general survey of security topics for FPGAs is given by Drimer [14].

### 3 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 4, we introduce netlist IP core identification and watermarking methods where the verification is done by using the extracted 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 [15] and [16] 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 5

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 [17]. Here, reading the watermark from an FPGA may take up to 10 minutes.

More about the different verification strategies can be found in [18].

## 4 Watermark Verification using the FPGA Bitfile

This section gives an overview of methods where the verification is done by *extracting an FPGA bitfile*. The bitfile can be analyzed to detect structures that can carry a watermark or that can be used to identify an IP core. Here, *lookup table contents* are used which are excellently suitable for watermarking and IP core identification. We start out by discussing how the contents of the lookup tables may be extracted from the FPGA bitfile. Following, methods for netlist IP core identification and watermarking are proposed (see also [19] and [20]).

### 4.1 Lookup Table Content Extraction

For FPGA designs, the functional lookup tables are an ideally suited component for carrying watermarks or using it for IP core identification. 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 [19], for example, a standard black-box reverse engineering procedure is applied to interpret Xilinx Virtex-II and Virtex-II Pro bitfiles.

## 4.2 Identification of Netlist Cores by Analysis of LUT Contents

In this approach, we do not add any signature or watermark. The core itself remains unchanged, so the functional correctness is given and no additional resources are used. We compare the content of the used lookup tables from the registered core with the used lookup tables in an FPGA design from the product of the accused company. If a high percentage of identical content is detected, the probability that the registered core is used is very high.

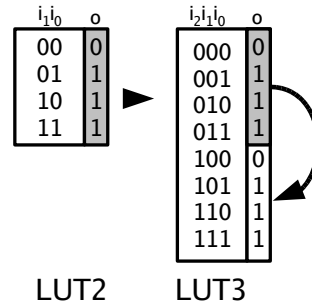
The synthesis tool maps the combinatorial logic of an FPGA core to lookup tables and writes these values into a netlist. After the synthesis step, the content of the lookup tables of a core is known, so we can protect netlist cores which are delivered at the logic level. The protection of bitfile cores at the device level is also possible.

After the core is purchased, the customer can combine this core with other cores. In the following CLB mapping step, it is possible that lookup tables are merged across the core boundaries or are removed by an optimizing transformation. This happens when different cores share logic or when outputs of the core are not used. These lookup tables cannot be found in the FPGA bitfile, but experimental results in [19] show that the percentage of these lookup tables compared to the number of all lookup tables in the core is typically low for the used mapping tool (Xilinx *map*).

After the extraction of the content of lookup tables from a bitfile, we can compare the obtained values with the information in the netlist. Unfortunately, the mapping tools do not necessarily adopt these values. The mapping tool may merge lookup tables from different cores together, convert one, two or three input lookup tables to four input lookup tables and permute the inputs to achieve a better routing.

All lookup tables of an FPGA have  $n$  inputs. On most FPGA architectures, lookup tables have  $n = 4$  inputs. In a core netlist, also lookup tables with less than  $n$  inputs may exist. These lookup tables must be mapped onto  $n$  input lookup tables. If one input is unused, only half of the memory is needed to store the function and the remaining space must be filled. In the case that a function uses less inputs than the underlying technology of the FPGA provides, it is desirable to turn the unused inputs into don't cares. Intuitively, this can be achieved rather easily by replicating the function table as it is demonstrated in Figure 2.

The mapping tool can permute the inputs of the lookup tables, for example, to achieve a better routing. In most FPGA architectures, the routing resources for lookup table inputs are not equal, and so a permutation of the lookup table inputs can lower the amount of used routing resources. Permutation of the inputs significantly alters the content of a lookup table. For  $n$  inputs,  $n!$  permutations exist and thus up to  $n!$  different lookup table values for one so-called *unique function*. To compare the contents of the lookup table from the netlist and the bitfile, it must be checked if one of these possible different lookup table values for one unique function is equal to the value of the lookup table in the bitfile.



**Fig. 2.** Converting a two input lookup table into a three input lookup table with unused input  $i_2$ .

This is done by creating a table with all possible values of lookup tables for all unique functions (see Figure 3).

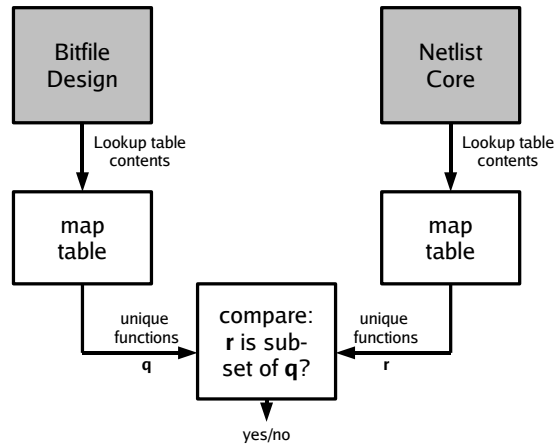
More about this method as well as experimental results and a robustness analysis can be found in [19] and [18]. The experimental results show that it is possible to identify a core in a design with a high probability.

### 4.3 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 automated 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



**Fig. 3.** Before the lookup table contents of the bitfile and the netlist are compared, they are mapped into unique functions.

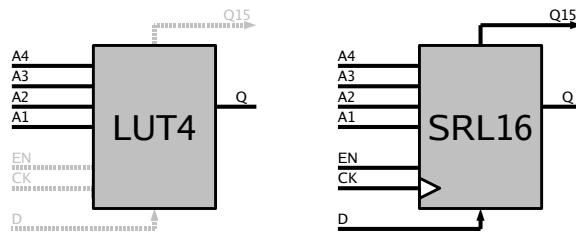
four inputs whereas the Virtex-5 has lookup tables with six inputs. However, in common netlist cores many logical lookup tables exist, which have less inputs than the type used on the FPGA.

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.

If a 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. Now, the extracted lookup table or shift register content can be used for a watermark detector which can decide if the watermark is embedded in the work or not.

A detailed description of this method as well as the experimental verification results and the overhead analysis are described in [20] and [18].





**Fig. 4.** In the Xilinx Virtex architecture, the same standard cell is used as a lookup table (LUT4) and also as a 16-bit shift-register lookup table(SRL16).

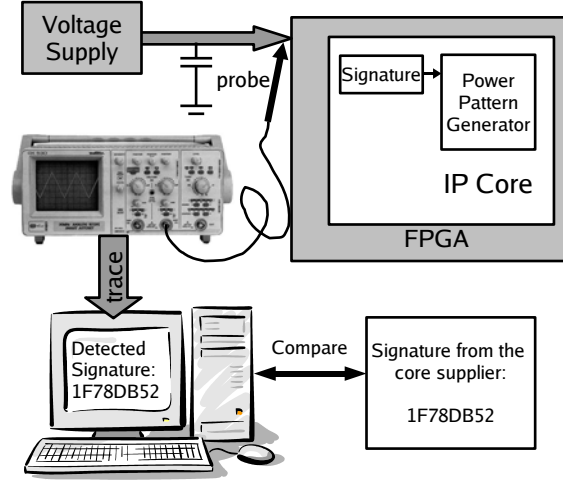
## 5 Power Watermarking

This section describes watermarking techniques introduced in [15] and [16], where a signature is verified over the *power consumption pattern* of an FPGA. For power watermarking methods, the term *signature* refers to the 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 consumption 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 5).

In the power watermarking approach described in [21] and [15], 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  $\{\sigma_0, \sigma_1\}^n$  [16] 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  $\sigma_i$  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  $\gamma$  is *power consumption through a shift*



**Fig. 5.** 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.

*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. The basic method with encoding scheme:  $\mathcal{Z} = \{(\gamma, 1, 1), (\bar{\gamma}, 1, 1)\}$ , the enhanced robustness encoding:  $\mathcal{Z} = \{(\gamma, 1, 32), (\bar{\gamma}, 1, 32)\}$ , and the BPSK approach:  $\mathcal{Z} = \{(\gamma, 1, \omega), (\bar{\gamma}, 1, \omega)\}$  are explained in detail in [15]. The correlation method with encoding  $\mathcal{Z} = \{(\gamma, 25, 1), (\bar{\gamma}, 25, 1)\}$  can be reviewed in [16]. 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 4.3, a shift register can also be used as a lookup table and vice versa in many FPGA architectures (see Figure 4 in Section 4.3). 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 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 one 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 [22].

## 6 Summary

In this paper, we have presented an overview of new approaches for identification and 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 concentrated 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. All methods mentioned in this overview paper are described in detail with experimental results in [18] and in the corresponding referenced papers.

## References

1. Boney, L., Tewfik, A.H., Hamdy, K.N.: Digital Watermarks for Audio Signals. In: International Conference on Multimedia Computing and Systems. (1996) 473–480
2. Lach, J., Mangione-Smith, W.H., Potkonjak, M.: Signature Hiding Techniques for FPGA Intellectual Property Protection. In: ICCAD '98: Proceedings of the 1998 IEEE/ACM international conference on Computer-aided design, New York, NY, USA, ACM (1998) 186–189
3. Saha, D., Sur-Kolay, S.: Fast Robust Intellectual Property Protection for VLSI Physical Design. In: ICIT '07: Proceedings of the 10th International Conference on Information Technology, Washington, DC, USA, IEEE Computer Society (2007) 1–6
4. Fan, Y.C., Tsao, H.W.: Watermarking for Intellectual Property Protection. *Electronics Letters* **39** (2003) 1316–1318
5. Kahng, A.B., Lach, J., Mangione-Smith, W.H., Mantik, S., Markov, I.L., Potkonjak, M.M., Tucker, P.A., Wang, H., Wolfe, G.: Constraint-Based Watermarking Techniques for Design IP Protection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **20** (2001) 1236–1252
6. Kirovski, D., Potkonjak, M.: 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)
7. Kirovski, D., Hwang, Y.Y., Potkonjak, M., Cong, J.: Intellectual Property Protection by Watermarking Combinational Logic Synthesis Solutions. In: ICCAD '98: Proceedings of the 1998 IEEE/ACM international conference on Computer-aided design, New York, NY, USA, ACM (1998) 194–198
8. Kahng, A.B., Mantik, S., Markov, I.L., Potkonjak, M.M., Tucker, P.A., Wang, H., Wolfe, G.: Robust IP Watermarking Methodologies for Physical Design. In: DAC '98: Proceedings of the 35th annual Design Automation Conference, New York, NY, USA, ACM (1998) 782–787
9. Bai, F., Gao, Z., Xu, Y., Cai, X.: A Watermarking Technique for Hard IP Protection in Full-custom IC Design. In: International Conference on Communications, Circuits and Systems (ICCCAS 2007). (2007) 1177–1180
10. Adelsbach, A., Rohe, M., Sadeghi, A.R.: Overcoming the Obstacles of Zero-knowledge Watermark Detection. In: MM&Sec '04: Proceedings of the 2004 workshop on Multimedia and security, New York, NY, USA, ACM (2004) 46–55
11. Li, Q., Chang, E.C.: Zero-knowledge Watermark Detection Resistant to Ambiguity Attacks. In: MMSec '06: Proceedings of the 8th workshop on Multimedia and security, New York, NY, USA, ACM (2006) 158–163
12. Abdel-Hamid, A.T., Tahar, S., Aboulhamid, E.M.: A Survey on IP Watermarking Techniques. *Design Automation for Embedded Systems* **9** (2004) 211–227
13. Ziener, D., Teich, J.: 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 (2005)
14. Drimer, S.: Security for Volatile FPGAs. (2009)
15. Ziener, D., Teich, J.: Power Signature Watermarking of IP Cores for FPGAs. *Journal of Signal Processing Systems* **51** (2008) 123–136

16. Ziener, D., Baueregger, F., Teich, J.: 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). (2010) 237–244
17. Kean, T., McLaren, D., Marsh, C.: 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, Washington, DC, USA, IEEE Computer Society (2008) 59–64
18. Ziener, D.: Techniques for Increasing Security and Reliability of IP Cores Embedded in FPGA and ASIC Designs. PhD thesis, University of Erlangen-Nuremberg, Germany (2010)
19. Ziener, D., Afmus, S., Teich, J.: Identifying FPGA IP-Cores based on Lookup Table Content Analysis. In: Proceedings of 16th International Conference on Field Programmable Logic and Applications (FPL 2006), Madrid, Spain (2006) 481–486
20. Schmid, M., Ziener, D., Teich, J.: Netlist-Level IP Protection by Watermarking for LUT-Based FPGAs. In: Proceedings of IEEE International Conference on Field-Programmable Technology (FPT 2008), Taipei, Taiwan (2008) 209–216
21. Ziener, D., Teich, J.: FPGA Core Watermarking Based on Power Signature Analysis. In: Proceedings of IEEE International Conference on Field-Programmable Technology (FPT 2006), Bangkok, Thailand (2006) 205–212
22. Ziener, D., Baueregger, F., Teich, J.: Multiplexing Methods for Power Watermarking. In: Proceedings of the IEEE Int. Symposium on Hardware-Oriented Security and Trust (HOST 2010), Anaheim, USA. (2010)