Verifying the Authorship of Embedded IP Cores: Watermarking and Core Identification Techniques

Jürgen Teich and Daniel Ziener
Hardware/Software Co-Design, Department of Computer Science
University of Erlangen-Nuremberg, Germany
email: *{juergen.teich, daniel.ziener}@cs.fau.de*

Abstract— *In this paper, we present an overview of existing watermarking techniques for FPGA and ASIC designs as well as our new watermarking and identification techniques for FPGA IP cores. Unlike most existing watermarking techniques, the focus of our new 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. Today, there is a huge market and repertoire of IP cores which can be seen in special aggregation web sites, for example [1] and [2], which administrate IP core catalogs.

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. Obviously, such concepts need advanced verification techniques to detect a signature or certain IP core characteristics, present in one of many IP cores inside a system. Furthermore, the embedded author identification should be preserved even when the IP cores pass through different design flow steps. 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 very high skilled employees assigned with the task to remove or bypass the embedded watermark.

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

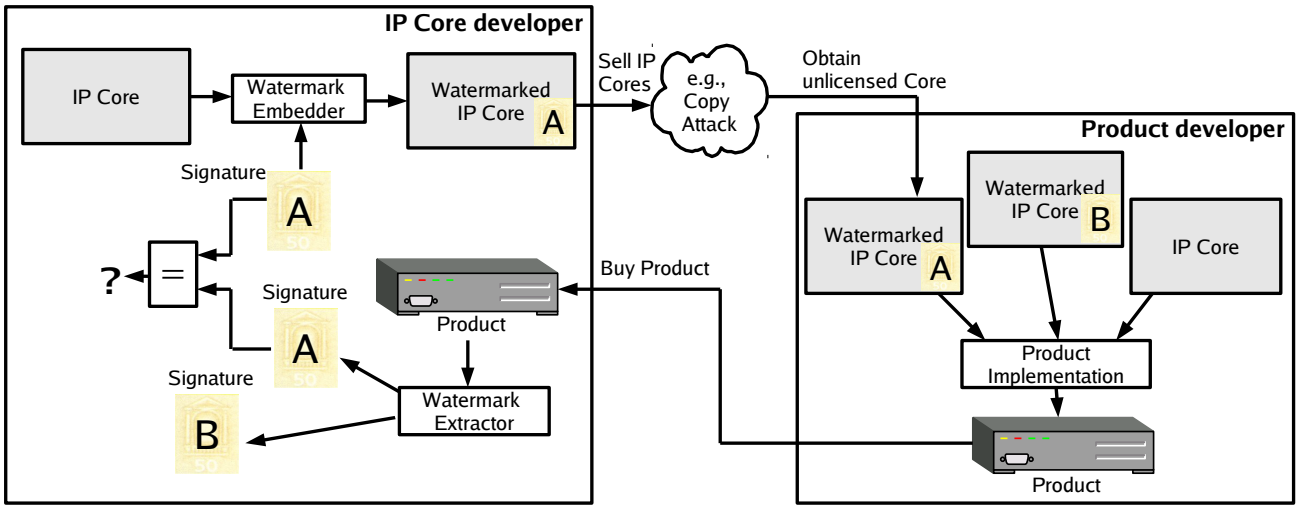

Fig. 1: This figure shows a typical watermarking flow: An IP core developer embeds a watermark A inside his core. If a product developer obtains an unlicensed core and embeds this core in his product, the IP core developer can buy this product and extract the watermarks of all used IP cores. Now, he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

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 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, an 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.

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 [3].

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. A *fingerprint* denotes a watermark which is varied for individual copies of a core. This technique can be used to identify individual authorized users. In case of an unauthorized copy, the user, the copied source belongs to, can be detected and the copyright infringement may be reconstructed. Watermarking procedures can be categorized into two groups of methods: *additive methods* and *constraint-based methods*.

A survey and analysis of watermarking techniques in the context of IP cores is provided by Abdel-Hamid and others [4]. Further, we refer to our own survey of watermarking techniques for FPGA designs [5]. Moreover, a general survey of security topics for FPGAs is given by Drimer [6].

2.1 Additive Watermarking of IP Cores

Additive methods are watermarking procedures, where a signature is added to the core. This means that the watermark is not embedded into the function of the core. Nevertheless, the watermark can be masked, so it appears to be part of the

functional part. Additive watermarks can be embedded into HDL, bitfile or layout cores.

2.1.1 HDL Cores

Additive watermarking for HDL cores seems to be very complicated, because of the human-readable structure of the HDL code. Hiding a watermark there is very difficult, because on the one hand, an attacker may easily detect the watermark, and on the other hand, subsequently used design tools might remove the watermark during circuit optimization. However, it is not impossible to include an additive HDL component into the core, which may not removed by the design tools.

Castillo et al. hide a signature into unused space of dedicated lookup table based memory [7]. To extract the signature, an additional logic monitors the input stream for a special *signature extraction sequence*. If this sequence is detected, the signature is sent to the outputs of the core. This approach was later generalized for other memory structures in [8].

Oliveira presents a general method for watermarking *finite state machines* (FSMs) in a way that on occurrence of a certain input sequence, a specific property exhibits [9]. The certain input sequence corresponds to the signature which is previously processed by cryptographic functions. A similar approach is presented by Torunoglu and others in [10] which explores unused transitions.

The disadvantage of these approaches is the usage of ports for signature verification. This works only if the ports are reachable. If the core is embedded into other cores, the ports of the watermarked core can be altered which falsifies or prevents the detection of the signature in the output stream. This applies also to the signature extraction sequence in the input stream.

2.1.2 Bitfile Cores

The approach of Lach and others watermarks bitfile cores by encoding the signature into *unused lookup tables* [11]. At first, the signature will be hashed and coded with an error correction code (ECC) to be able to reconstruct the signature even if some lookup tables are lost, e.g., during tampering. After the initial place and route pass, the number of unused lookup tables will be determined. The signature is split into the size of the lookup tables and additional LUTs are added to the design. Then, the place and route process will be started again with the watermarked design. Later, the approach was improved by using many small watermarks instead of a single large one [12]. The size of the watermarks should be limited by the size of a lookup table. The advantage is that small watermarks are easier to search for, and for verification, only a part of all of watermark positions must be published. With the knowledge of the published position, the watermark can be easily removed by an attacker. At the verification process, only a few positions of the watermark need to be used to establish the ownership. A second improvement is that a *fingerprinting technology* is added to the approach that enables the owner to see which customer has given the core away [13]. The fingerprinting technology is achieved by dividing the FPGA into tiles. In each tile, one lookup table is reserved for the watermark. The position of the mark in the tile encodes the fingerprint. For verification, it is possible to read out the content of the lookup table from a bitfile. So, these methods are easy to verify. It's more difficult to determine the position of the watermark in a tile, but it's still generally possible. However, if an attacker knows the position of the watermark, it is easy to overwrite it.

Saha and others present a watermarking strategy for FPGA bitfiles by subdividing the lookup table locations into sets of 2×2 tiles [14]. The number of used lookup tables in a set is used as signature. From an initial level, additional lookup tables are added to achieve the fill level according to the signature. The input and output are connected to the *don't care inputs* of the neighboring cells. Kahng and others show in [15] that the configuration of the multiplexer of unused CLB outputs in FPGA bitfiles can carry a signature. The signature is embedded after the bitfile creation and by knowing the encoding of the bitfile. These configuration bits can be later extracted to verify the signature.

Van Le and Desmedt show that these additional watermark schemes for bitfile cores can be easily attacked by reverse engineering, watermark localization, and subsequent watermark removal [16]. A simple algorithm is introduced which identifies lookup tables or multiplexers whose outputs are not connected to any output pins. However, these attacks are only successful if reverse engineering of the bitfile is possible and the costs of reverse engineering are not too high.

Finally, Kean and others present a watermarking strategy where a signature is embedded into an FPGA bitfile core or design [17]. The read out of the signature is done by measuring the temperature of the FPGA. This approach is commercially available as the product *DesignTag* from *Algotronix*.

2.2 Constraint-Based Watermarking of IP Cores

All optimization problems have constraints which must be satisfied to achieve a valid solution. Solutions which satisfy this constraints are the *solution space*. Constraint-based watermarking techniques represent a signature as a set of *additional constraints* which are applied to the hardware optimization and synthesis problem. These additional constraints reduce the solution space since the chosen solution must also satisfy the additional constraints [18], [19].

Qu proposes a methodology to make a part of the watermark – for constraint-based watermarking, some additional constraints – public which should deter attackers [20]. The other parts, called *private watermark*, are only known by

the core author and are used to verify the authorship in case that the public watermark was attacked. A similar approach is used by Qu and others to generate different fingerprints by dividing the additional constraints into two parts [21]: The first part is a set of relaxed constraints which denote the watermark. By applying distinct constraints to the second part, different independent solutions can be generated which may be used as diverse fingerprinted designs.

Charbon proposed a technique to embed watermarks on different abstraction levels which he called *hierarchical watermarking* [22]. The idea is, if an attacker is able to remove a watermark, for example, embedded into the layout of a circuit, the watermarks added at higher abstraction levels are still present. However, Charbon focused more on *layout, nets*, and *latch watermarking techniques* which are only applicable for ASIC layout cores.

The verification of a constraint-based watermark is usually done with the watermarked core as it is. This means the watermarked core can be purchased or published and from the distributed cores the watermark can be verified. However, if the core is combined with other cores and traverses further design steps, the watermark information is usually lost or it cannot be extracted.

Van Le and Desmedt [16] present an ambiguous attack for constraint-based watermarking techniques. The authors add further constraints to the watermarked solution by allowing only a minimal increase of the overhead. The result is a slightly degenerated solution which satisfies many additional constraints. This means that in this solution, a lot of different signatures can be found which destroys the unique identification of the core developer. They choose, for example, the constraint-based watermarking approach for *graph coloring*. Further, this attack might be applicable to other constraint-based watermarking techniques.

As it was the case with additive watermarking strategies, constraint-based watermarking strategies are applicable for HDL, netlist, and bitfile cores.

2.2.1 HDL Cores

HDL code is usually produced by human developers or high-level synthesis tools. Both can set additional constraints to watermark a design. One approach is to use a watermarked *scan chain* [23]. Scan chains are usually used in ASIC designs to access the internal registers for debugging purposes. The use of scan chains in FPGA designs is rather unusual, but might be helpful in some cases. Depending on the signature, we have a variation on the scan chains which can be used to detect the watermark. This approach is easy to verify, if the scan chains can be accessed from outside of the chip. Problems occur, if the scan chain is only used internally or is not connected to any device. In such a case, there is no verification possibility.

Some work was done for watermarking *digital signal processing* (DSP) functions [24], [25]. This kind of watermarking has more in common with media watermarking instead if IP watermarking. Both approaches alter the function of the core slightly by embedding a watermark. In [24], the coefficients of *finite impulse response* (FIR) filters are slightly varied according to the watermark. Additionally, the authors use different structures to build the FIR filter which also corresponds to the signature. In [25], these ideas are extended and proven correct by mathematical analysis.

2.2.2 Netlist Cores

An approach to watermark netlist cores is to preserve certain nets during *synthesis* and *mapping* [19]. Synthesis tools merge signals or nets together and produce new nets. Only a few nets from the synthesis input will be visible in the synthesis result. The technology mapping tool also eliminates nets by assembling gates together in a lookup table. Kirovski's approach enumerates and sorts all nets in a design. The first nets of the input are chosen by the synthesis tools according to a signature. These nets will be prevented from elimination by the design tools by connecting these nets to a temporary output of the core. The new outputs from additional constraints for the synthesis tool, and the corresponding result is related to the watermark. A disadvantage is that it is easy to remove the additional logic. If the content of the lookup table is synthesized again, the watermark will be removed.

Meguerdichan and others presented a similar approach for netlist cores where additional constraints are added during the *technology mapping step* of the synthesis process [26]. In this approach, critical signals are not altered which preserves the timing and the performance of the core. The signature is encoded into the number of allowed inputs of a certain primitive cell, e.g., a gate or a lookup table. The primitive cells which are not in the critical path are enumerated, and according to the signature, the number of usable inputs are constrained.

Khan and others watermark netlist cores by doing a *rewiring* after synthesis [27]. Rewiring means that redundant connections between primitive cells are added in the netlist which makes other original connections redundant. These new redundant connections are removed.

Bai and others introduce a method for watermarking transistor netlists for *full custom designs* [28]. The transistors are enumerated and sorted into a list like in the approach above. Corresponding to the pseudo random stream generated from the signature, the width of the transistor gate is altered. If the transistor is assigned a '1' from the random stream, the transistor width is increased by a constant value.

2.2.3 Bitfile and Layout Cores

Additional placement, routing, or timing constraints can be added to watermark bitfile cores. To embed a watermark with placement constraints, Kahng and others place the *configurable logic blocks* (CLBs) in even or odd rows depending on the signature [29]. In this approach, the signature is

transformed into even/odd row placement constraints. The placed core will be tested on preserving the constraints and, if necessary, CLBs are swapped. The problem of verification is to extract the CLB placement information. Only if knowing how the CLBs correspond to the signature, the watermark can be verified. A strategy to achieve this is to uniquely enumerate the CLBs in an FPGA from the top left corner.

Kahng and others [29] propose a second approach by adding constraints to the router. The constraints achieve that a net selected by the signature is routed with some additional, unusual routing resources. These unusual resources can be, for example, *wrong way* segments. A wrong way segment is a segment in which the net goes to the wrong direction and then back in the right direction to form a backstrap. The authors claim that this is unlikely for a normal router, and so such a net can be verified as a watermarked net.

Saha and others present a watermarking scheme by altering the size of the repeaters according to the signature [14]. In high performance ASIC designs, *repeaters* (a buffer for amplification of the signal) are inserted into critical nets to decrease the delay.

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 [30] and [31] 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 [32].

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 and IP core identification are proposed (see also [33]). Following, watermarking methods for bitfile and netlist cores are discussed (see also [34]). The focus of these watermarking methods lies on the usage of *functional lookup tables* in order to increase the robustness against removal attacks. The term *functional lookup table* refers to lookup tables which are already used in a given (non-watermarked) IP core and represent a part of the functional logic of the core which may not be removed by an attacker in order to retain the correctness of the core.

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 [33], 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 [33] 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 $n = 4$ or $n = 6$ 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.

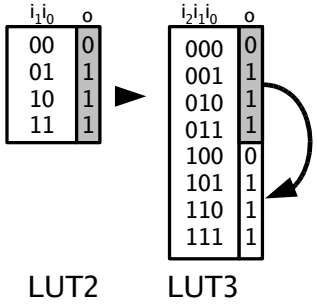

Fig. 2: Converting a two input lookup table into a three input lookup table with unused input i_2.

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. 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

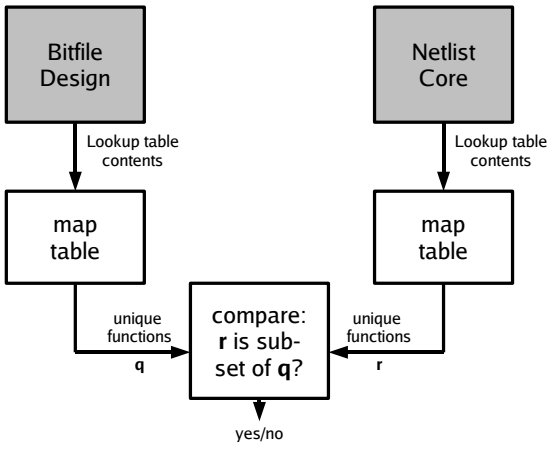

Fig. 3: Before the lookup table contents of the bitfile and the netlist are compared, they are mapped into unique functions.

and a robustness analysis can be found in [33] and [32]. The experimental results show that it is possible to identify a core in a design with a high probability.

4.3 Identification of HDL Cores by Analysis of LUT Contents

In the last section we have shown that is possible to identify an IP core, distributed as a netlist, in an FPGA design by analyzing the LUT contents of the configuration bitfile. However, many IP cores are published at the RTL abstraction level as HDL core.

To identify HDL cores, the lookup table contents can be used as well. However, the lookup table content is generated by the synthesis step, which is executed after the publication of the HDL cores. Therefore, a pirate who can obtain an unlicensed HDL core, controls the complete design flow from the RTL to the device level. It is up to the pirate to decide which synthesis tool is used to synthesize the core and therefore, create the lookup table contents. Different synthesis tools might create different lookup table contents. To prove or disprove this assumption, we analyzed common synthesis tools with respect to the generation of lookup table contents.

The first step is to analyze different netlist cores to find out whether they were generated from the same HDL core. The goal is to find different netlist cores which can be assigned to a corresponding HDL source even if they were synthesized with different tools and different synthesis parameters.

The comparision is based on the lookup table contents. Therefore, to compare two netlist cores, the first step is to extract the lookup table contents from the netlist cores and map these to unique functions. The probability that both cores were generated from the same source is high if a high percentage of lookup tables which can be found in both cores implement in both cores the same unique functions. A detailed description of this method as well as experimental results are presented in [32].

This method is the first steps towards an identification of HDL cores in bitfiles. Here, we concentrated on the synthesis step between the RTL and logic abstraction level. However, to build the complete chain for identification of HDL cores from bitfiles some links are missing. Identifying HDL cores in netlists has an inherent uncertainness comparable to the identification of netlist cores in bitfiles. By combining both techniques the uncertainness can be too high to give a trustful result. Nevertheless, this is an interesting topic for future research.

4.4 Watermarks in LUTs for Bitfile Cores

In this section, we introduce our first watermarking technique for IP cores. The easiest way to watermark an FPGA design is to place the watermarks into the bitfiles. Bitfiles are very inflexible because they were specifically generated for a certain FPGA device type, however, it makes sense to sell bitfile IP cores for common development platforms which carry the same FPGA type. Usually, a bitfile core is a whole design which is completely placed and routed and therefore ready to use. There also exist partial bitfiles which carry only one core. These partial bitfile cores can be combined into one FPGA which increases the flexibility of these cores and therefore may increase the trade possibilities.

In this approach, we hide our signature inside unused lookup tables. It is very unlikely that a design or bitfile core uses all available lookup tables in an FPGA. Before a design reaches this limit, the routing resources are exhausted and the timing degenerates rapidly. Therefore, many unused lookup tables exist in usual designs. On the other hand, lookup table content extraction is not difficult. Using lookup tables for hiding a watermark which are far away from the used ones, makes it easier for an attacker to identify and remove them. Even if an attacker is able to extract all lookup tables from a bitfile core, the lookup tables which carry the watermark should not be suspicious.

In Xilinx devices, lookup tables are grouped together with *flip-flops* into slices. A slice usually consists more than one lookup table, e.g., the Virtex-II and Virtex-II Pro devices have two lookup tables in one slice. It is not unusual that only one lookup table of a slice is used and the other remains unused. Hiding a watermark in the unused lookup table of a used slice is less obvious than using lookup tables in unused slices. Even if the attacker is able to extract the lookup table content and coordinates, the watermarks are hard to detect.

The extraction and verification of the watermark is rather easy. First of all, the content and the coordinates of all used lookup table of the core are extracted. For the verification there exist two approaches: a *blind approach* and a *non-blind approach*. In the blind approach, the watermarks are searched in all extracted lookup table contents, whereas in the non-blind approach the location of the watermarks are known. Having the right coordinates, the watermarked lookup table content can be directly compared to the watermarks of the core developer. The locations of the watermarks delivered from the core developer, however, should be kept secret, because otherwise it is very easy for an attacker to remove the marks.

More about this method as well as experimental results and a robustness analysis can be found in [32].

4.5 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.

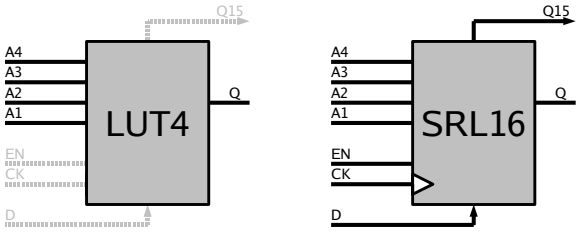

Fig. 4: In the Xilinx Virtex architecture, the same standard cell is used as a lookup table (LUT4) and also as a 16-bit shift-register lookup table (SRL16).

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.

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 [34] and [32].

5. Power Watermarking

This section describes watermarking techniques introduced in [30] and [31], 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 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 5).

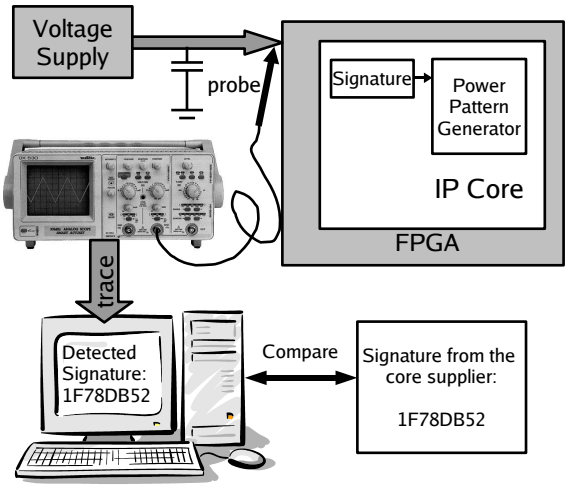

Fig. 5: Watermark verification using power signature analysis: From a signature (watermark), a power pattern inside the core will be generated that can be probed at the voltage supply pins of the FPGA. From the trace, a detection algorithm verifies the existence of the watermark.

In the power watermarking approach described in [35] and [30], 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 $\{\sigma_0, \sigma_1\}^n$ [31] 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. 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 [30]. The correlation method with encoding $\mathcal{Z} = \{(\gamma, 25, 1), (\bar{\gamma}, 25, 1)\}$ can be reviewed in [31]. 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. For example, the typical swing of an FPGA core voltage signal which results from a shift of a huge shift register is shown in Figure 6.

As mentioned before in Section 4.5, a shift register can also be used as a lookup table and vice versa in many FPGA architectures (see Figure 4 in Section 4.5). 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

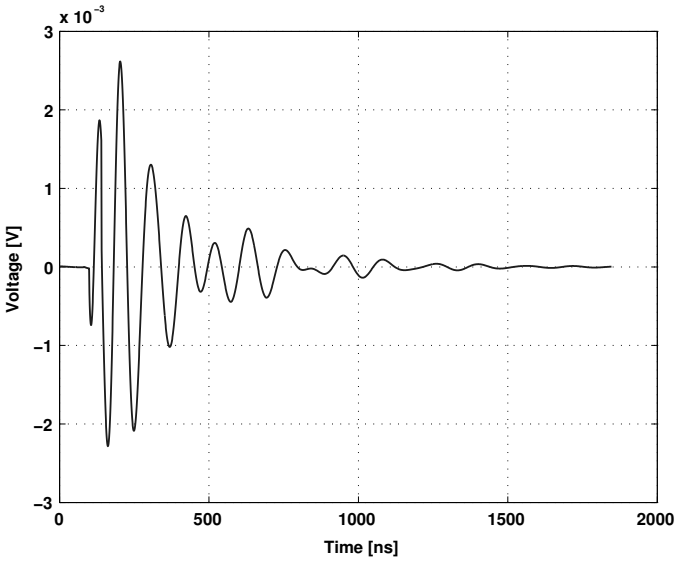

Fig. 6: This measurement is obtained by a shift of a huge shift register, implemented using 128 SRL16 primitive cells in the Spartan-3 FPGA on the Digilent Spartan-3 starter board [36]. Note that the DC component of the core voltage signal is removed by an AC filter.

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 4.5 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 consists of two steps. First, the core 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 (see Figure 7).

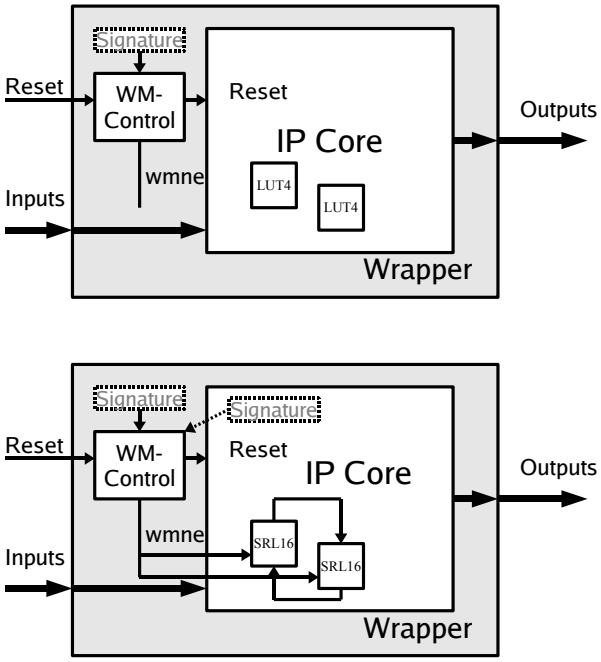

Fig. 7: The core and the wrapper before (above) and after (below) the netlist alternation step. The signal “wmne" is an enable signal for shifting the power pattern generator shift register.

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 embedding procedure for Virtex-II netlist cores is done by a program which parses an EDIF netlist and writes back the modified EDIF netlist. First, the program reads all LUT4 instances. Then, the instances are converted to a shift register (SRL16), if required, initialized with the shifted value and connected to the clock and the watermark enable (wmne) signal according to Figure 7. Always two shift registers are connected together to rotate their contents. Finally, the modified netlist is created. The watermarked core is now ready for purchase or publication.

A company may obtain an unlicensed version of the core and embeds this core in a product. 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. The detecting function depends on the encoding scheme. In [30] and [31], 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 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 [37]. In this paper we show that the most promising techniques for use on an FPGA are time (TDM) and code (CDM) multiplexing.

6. Summary

In this paper, we have presented an overview of existing and new approaches for identication 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 [32] and in the corresponding referenced papers.

References

[1] Design & Reuse, "Catalyst of Collaborative IP Based SoC Design," *URL: http://www.design-reuse.com/*.

[2] Chip Estimate, "ChipEstimate.com," *URL: http://www.chipestimate.com/*.

[3] L. Boney, A. H. Tewfik, and K. N. Hamdy, "Digital Watermarks for Audio Signals," in *International Conference on Multimedia Computing and Systems*, 1996, pp. 473–480. [Online]. Available: citeseer.ist.psu.edu/boney96digital.html

[4] A. T. Abdel-Hamid, S. Tahar, and E. M. Aboulhamid, "A Survey on IP Watermarking Techniques," *Design Automation for Embedded Systems*, vol. 9, no. 3, pp. 211–227, 2004.

[5] D. Ziener and J. Teich, "Evaluation of Watermarking Methods for FPGA-Based IP-cores," University of Erlangen-Nuremberg, Department of CS 12, Hardware-Software-Co-Design, Am Weichselgarten 3, D-91058 Erlangen, Germany, Tech. Rep. 01-2005, Mar. 2005.

[6] S. Drimer, "Security for Volatile FPGAs," Nov. 2009.

[7] E. Castillo, L. Parrilla, A. Garcia, A. Loris, and U. Meyer-Baese, "IPP Watermarking Technique for IP Core Protection on FPL Devices," in *International Conference on Field Programmable Logic and Applications, 2006. FPL'06*, 2006, pp. 487–492.

[8] E. Castillo, L. Parrilla, A. Garcia, U. Meyer-Baese, G. Botella, and A. Lloris, "Automated Signature Insertion in Combinational Logic Patterns for HDL IP Core Protection," in *4th Southern Conference on Programmable Logic, 2008*, 2008, pp. 183–186.

[9] A. L. Oliveira, "Techniques for the Creation of Digital Watermarks in Sequential Circuit Designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 9, pp. 1101–1117, 2001.

[10] I. Torunoglu and E. Charbon, "Watermarking-based Copyright Protection of Sequential Functions," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 3, pp. 434–440, 2000.

[11] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "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, pp. 186–189.

[12] J. Lach, W. H. Mangione-Smith, and Potkonjak, "Robust FPGA Intellectual Property Protection through Multiple Small Watermarks," in *DAC '99: Proceedings of the 36th annual ACM/IEEE Design Automation Conference*. New York, NY, USA: ACM, 1999, pp. 831–836.

[13] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "Fingerprinting Techniques for Field-Programmable Gate Array Intellectual Property Protection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. volume 20, 2001.

[14] D. Saha and S. Sur-Kolay, "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, pp. 1–6.

[15] A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. M. Potkonjak, P. A. Tucker, H. Wang, and G. Wolfe, "Watermarking Techniques for Intellectual Property Protection," in *DAC '98: Proceedings of the 35th annual Design Automation Conference*. New York, NY, USA: ACM, 1998, pp. 776–781.

[16] T. V. Le and Y. Desmedt, "Cryptanalysis of UCLA Watermarking Schemes for Intellectual Property Protection," in *IH '02: Revised Papers from the 5th International Workshop on Information Hiding*. London, UK: Springer-Verlag, 2003, pp. 213–225.

[17] T. Kean, D. McLaren, and C. 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*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 59–64.

[18] A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. M. Potkonjak, P. A. Tucker, H. Wang, and G. Wolfe, "Constraint-Based Watermarking Techniques for Design IP Protection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 10, pp. 1236–1252, 2001.

[19] D. Kirovski, Y.-Y. Hwang, M. Potkonjak, and J. 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*. New York, NY, USA: ACM, 1998, pp. 194–198.

[20] G. Qu, "Publicly Detectable Watermarking for Intellectual Property Authentication in VLSI Design," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 21, no. 11, pp. 1363–1367, 2002.

[21] G. Qu and M. Potkonjak, "Fingerprinting Intellectual Property using Constraint-addition," in *DAC '00: Proceedings of the 37th Annual Design Automation Conference*. New York, NY, USA: ACM, 2000, pp. 587–592.

[22] E. Charbon, "Hierarchical Watermarking in IC Design," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, 1998, pp. 295–298.

[23] D. Kirovski and M. 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. [Online]. Available: citeseer.ist.psu.edu/218548.html

[24] A. Rashid, J. Asher, W. H. Mangione-Smith, and M. Potkonj, "Hierarchical Watermarking for Protection of DSP Filter Cores," in *Proceedings of the Custom Integrated Circuits Conference. Piscataway, NJ*. IEEE Press, 1999, pp. 39–45.

[25] R. Chapman and T. S. Durrani, "IP Protection of DSP Algorithms for System on Chip Implementation," *IEEE Transactions on Signal Processing*, vol. 48, no. 3, pp. 854–861, 2000.

[26] S. Meguerdichian and M. Potkonjak, "Watermarking while Preserving the Critical Path," in *DAC '00: Proceedings of the 37th Annual Design Automation Conference*. New York, NY, USA: ACM, 2000, pp. 108–111.

[27] M. M. Khan and S. Tragoudas, "Rewiring for Watermarking Digital Circuit Netlists," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 7, pp. 1132–1137, 2005.

[28] F. Bai, Z. Gao, Y. Xu, and X. Cai, "A Watermarking Technique for Hard IP Protection in Full-custom IC Design," in *International Conference on Communications, Circuits and Systems (ICCCAS 2007)*, 2007, pp. 1177–1180.

[29] A. B. Kahng, S. Mantik, I. L. Markov, M. M. Potkonjak, P. A. Tucker, H. Wang, and G. Wolfe, "Robust IP Watermarking Methodologies for Physical Design," in *DAC '98: Proceedings of the 35th annual Design Automation Conference*. New York, NY, USA: ACM, 1998, pp. 782–787.

[30] D. Ziener and J. Teich, "Power Signature Watermarking of IP Cores for FPGAs," *Journal of Signal Processing Systems*, vol. 51, no. 1, pp. 123–136, April 2008.

[31] D. Ziener, F. Baueregger, and J. 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)*, May 2010, pp. 237–244.

[32] D. Ziener, "Techniques for Increasing Security and Reliability of IP Cores Embedded in FPGA and ASIC Designs," Dissertation, University of Erlangen-Nuremberg, Germany, July 2010, verlag Dr. Hut, Munich, Germany.

[33] D. Ziener, S. Aßmus, and J. 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)*, Madrid, Spain, Aug. 2006, pp. 481–486.

[34] M. Schmid, D. Ziener, and J. Teich, "Netlist-Level IP Protection by Watermarking for LUT-Based FPGAs," in *Proceedings of IEEE International Conference on Field-Programmable Technology (FPT 2008)*, Taipei, Taiwan, Dec. 2008, pp. 209–216.

[35] D. Ziener and J. Teich, "FPGA Core Watermarking Based on Power Signature Analysis," in *Proceedings of IEEE International Conference on Field-Programmable Technology (FPT 2006)*, Bangkok, Thailand, Dec. 2006, pp. 205–212.

[36] Digilent Inc., "Spartan-3 Starter Board," *URL: http://www.digilentinc.com/*.

[37] D. Ziener, F. Baueregger, and J. 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.