Hardware IP Watermarking and Fingerprinting

Chip-Hong Chang, Miodrag Potkonjak, and Li Zhang

Abstract The continuously increasing gap between silicon and designer productivity has created a need for common design reuse. The initial response to this need emphasized the enforcement of designers' rights through the creation of design such that ownership can be proved with ultra high probability. Our primary objective is to provide sound treatment of the foundations of hardware watermarking and fingerprinting, as well as key research contributions to the field. At the same time, we aim not just to survey and explain key ideas, concepts, and tools but also to identify dominating trends and briefly outline emerging hardware IP protection and, in particular, hardware watermarking and fingerprinting issues including the creation and validation of trusted hardware IP and semantic IP rights protection. Finally, we also elaborate on changing focus from techniques for embedding watermarks and fingerprints to approaches for watermark and fingerprint extraction and on remote digital rights enforcement of hardware IP rights.

1 Introduction

Watermarking of hardware designs is a procedure in which a signature of the designer is embedded into a design in such a way that the design's correct functionality is not impacted and all design metrics are minimally or not impacted at all. Watermarking should be conducted in such a way that watermark extraction is easy while its removal is very difficult in terms of the required design effort and the induced manufacturing and testing cost. In addition, several other requirements may be included such that proving the existence of the embedded signature can be accomplished without revealing it, that a watermarked design can be recognized

C.-H. Chang (✉) • L. Zhang
School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore
e-mail: echchang@ntu.edu.sg; lzhang2@e.ntu.edu.sg

M. Potkonjak
Computer Science Department, UCLA, Los Angeles, CA 90095-1596, USA
e-mail: miodrag@cs.ucla.edu

C.-H. Chang, M. Potkonjak (eds.), *Secure System Design and Trustable Computing*, DOI 10.1007/978-3-319-14971-4_10

when it is used within a larger design and that even when a subpart of the watermarked design is used the original author can prove his/her authorship.

The first hardware watermarking efforts were developed in 1997 and reported in early 1998 [1–3]. There were three main sources of inspiration for their development. The first was the emergence of watermarking approaches for audio and video artifacts due to the rapid growth of this market caused by the explosive growth of the Internet. The main idea behind media watermarking is to leverage imperfections in human audio and vision systems so that recorded signals preserve their subjective fidelity while the signatures that indicate the ownership are embedded into them. A great variety of audio, image, video, and text watermarking has been proposed, implemented, and evaluated. Media watermarking has been established as a research area by itself. In particular, a large number of patents have been issued.

The second starting point was the growing gap between silicon and design productivity. The reuse of hardware intellectual property was the best synthesis alternative. It was widely expected that hardware IP blocks will form a viable market. Thus, the Virtual Socket Interface Alliance (VSIA) was formed to coordinate the creation of viable and fast growing hardware IP markers and standards. One of the six main thrusts of VSIA was intellectual property protection (IPP), which resulted in the establishment of the first two hardware watermarking standards.

While the first two impetus sources were market driven, the last one was completely technical. Many synthesis tasks correspond to combinatorial optimization tasks. For example, register assignment in behavioral synthesis and embedded compilation correspond to graph coloring. It has been observed experimentally that for each of these tasks there exist numerous, sometimes even exponentially many, solutions of identical or very similar quality [4–10]. Numerous solutions with optimal and near optimal quality directly enable hardware IP watermarking because the designer can select as his watermarked design a solution with the special property that it serves as proof of legal ownership.

A closely related and extended problem is hardware IP fingerprinting, which is a procedure where a unique tag is created for each chip or a set of ICs. For instance, a design house can create for each of its customers an IP instance with a unique fingerprint for tracing which IP instance is resold by which customer. The main challenge is to avoid very expensive redesign steps for all consequent tasks and to eliminate or at least reduce the number of required unique masks.

The essential difference between watermarking media artifacts and hardware IP is that the latter is a functional entity and its functionality must be fully preserved. In addition, it is important to consider negative impacts on design metrics such as speed and energy. Hence, media watermarking techniques are not applicable. Two main directions have been proposed for hardware watermarking. The first is that additional constraints and/or modification of optimization objectives are imposed in such a way that the quality of the solution is minimally impacted while long and convincing proof of the ownership is guaranteed [11–13]. The second direction is that additional functionality is added as the watermark. For example, a state

transition graph is augmented by a specific set of transitions [14, 15], or the unit impulse response of a filter is over-specified in such a way that the author's signature can be observed [16].

The latter approach is interesting because it intrinsically provides for nondestructive and easy extraction of the watermark. That is, in general, a very important issue with hardware watermarking: it is important that remote and easy extraction of watermarking is provided. While hardware reverse engineering techniques are very advanced [17–21], in some applications, nondestructive and remote watermark extraction is essential.

Surprisingly, a common conceptual misconception is that watermarking solutions of synthesis steps is considered to be equivalent to watermarking of corresponding generic graph theoretic problems [22]. It is indeed a standard and widely used technique to map register assignment in behavioral synthesis to graph coloring theoretic frameworks and to use one of many available algorithms [23]. However, once the solution for a corresponding graph coloring problem is incrementally altered with intention of eliminating embedded watermarks, the overall design is also altered. For example, different control logic is required for deciding when which register is accessed as a read or a write. Furthermore, the new control logic requires new interconnects and new control signals from the finite state machine of the overall design. Hence, we need new global and local routing that may impact the overall timing that now has to be validated. In modern designs this may be a very expensive task due to factors such as process variation and crosstalk. Finally, the creation of new masks may be a very expensive proposition. So, for the effective watermarking removal step of graph coloring-based register assignment it is not sufficient to just change registers where variable placement corresponds to assigning corresponding nodes in the corresponding instance of graph coloring. In addition, one has to make sure that consequent synthesis and implementation steps do not have to be altered, which is very often a much more demanding and often impossible requirement to satisfy for designs of any realistic complexity.

Another common related misconception along a similar line of reasoning is that there is not much difference between hardware and software watermarking and that the same techniques can be used interchangeably in the two domains. While this observation is sometimes indeed true and the same concepts are applicable in both domains, there is an essential difference between software and hardware watermarking. However, as we just explained, hardware watermarking is greatly protected due to high synthesis and validation costs of the consequent tasks as well as with the time and economic expense of creating new masks for silicon manufacturing. On another hand additional compilation costs are rather low and often even negligible. Thus, for software watermarking one cannot count on implicit and intrinsic synthesis, validation, and manufacturing costs as effective attack deterrents.

2 Problem Formulation

2.1 SoC Design Process and IP Core Types

The SoC design process, as shown in Fig. 1, starts from constructing a system-level model based on the functional specifications of the system. The system-level model consists of interconnected functional blocks, each of which is assigned a specific software or hardware resource. The assignment creates behavioral models of the software and hardware parts of the system. The two parts are then developed separately before they are integrated together in the final stage to form a complete system.

Our discussion focuses on the hardware development. IP cores in different forms are developed and/or adopted in different abstraction levels of the IC design flow.

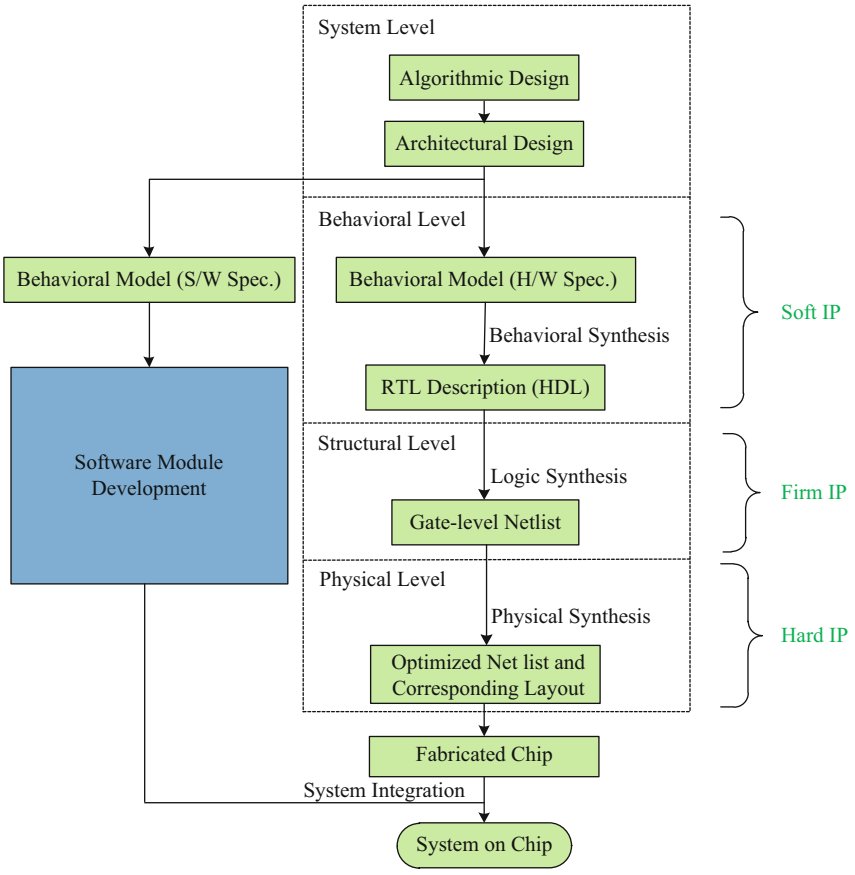

Fig. 1 SoC design flow and three main types of hardware IP cores

According to the Virtual Socket Interface (VSI) Alliance architecture document [24], there are three major types of IP cores, i.e., the soft, firm and hard IP cores.

The soft IP core is used in the behavioral level. These IP cores are usually delivered in the form of synthesizable hardware description language (HDL). This type of IPs provides excellent flexibility to match the requirements of a specific system. The drawback of these IP cores is that their performance is highly dependent on the optimization effort of the system integrator, which is less predictable than the other two types of IP cores.

The hard IP core is used in the physical level. These IP cores target a specific technology and are delivered in the form of fully optimized netlist or the corresponding physical layout. This type of IP cores offers the best performance for the chosen technology library; but due to process dependencies, they have much less flexibility and portability than the soft IP cores. Without requiring any further optimization, the hard IP core is generally released as a drop-in replacement in the physical level design of the system.

The firm IP, or semi-hard IP, refers to IP cores that are in an intermediate form between the soft IP and hard IP. They are usually delivered in the form of gate-level netlist in the structural level. This type of IP cores has more predictable performance than the soft IP and better flexibility and portability than the hard IP. They may also be optimized using a generic technology library, including even physical synthesis steps like floor planning and placement. Nonetheless, no routing is performed and the firm IP remains relatively technology independent. After integrating the firm IP into the system, the system integrator still needs to perform physical synthesis for optimization in a specific technology.

As the soft IP is provided to the system integrator in the form of synthesizable high-level source code, security risk of this IP type is the highest. The high flexibility for reuse also makes infringement of the soft IP very difficult to be detected and traced. In contrast, the limited flexibility of hard IP makes it the easiest to be protected among the three IP types.

The firm IP is sometimes delivered with the synthesizable register-transfer-level (RTL) code for the ease of reuse by the system integrator. In this case, the risk of the firm IP is as high as that of the soft IP. However, if it is transferred alone, i.e., only the gate-level netlist is provided, the firm IP faces the medium risk among the three types of IP cores. The characteristics of the three types of IP cores are summarized in Table 1.

Table 1 Characteristics of the three types of IP cores

	Soft IP	Firm IP	Hard IP
Abstraction level	Behavioral	Structural	Physical
Optimization	Low	Medium	High
Technology dependency	Independent	Generic	Dependent
Flexibility	High	Medium	Low
Distribution risk	High	High (with RTL); Medium (no RTL)	Low

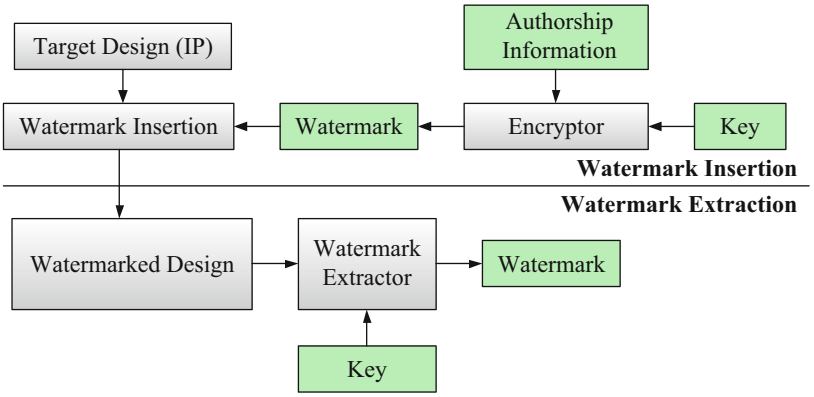

Fig. 2 Generic model for IP watermarking

2.2 Generic Model and Desiderate for IP Watermarking

Similar to watermarking techniques for multimedia applications (e.g., pictures, audio, video or 3D models), hardware IP watermarking is realized by inserting covert and indelible ownership information into the target design for ownership proof. Derived from the generic model for digital watermarking [25], a generic model for hardware IP watermarking is depicted in Fig. 2. The watermark insertion process inserts the watermark into an IP core at some chosen design abstraction level, while the watermark extraction process defines how the watermark is extracted from the watermarked IP core.

The general requirements for IP watermarking are very similar to those of multimedia watermarking. Nonetheless, multimedia watermarking has more freedom to alter the cover media and insert the watermark; it exploits human auditory or visual imperfections to achieve watermark imperceptibility and robustness. Such alteration is restricted in IP watermarking, since the watermarked IP must remain functionally correct. Based on the requirements for an IP watermarking scheme proposed in [26–28], the following desiderata are outlined:

Maintenance of Functional Correctness The functionality of the IP core should not be altered after the insertion of the watermark.

Independence of the Secrecy of Algorithm According to one of the oldest security tenets defined by Kerckhoffs [29], the security of any encryption or security technique lies not in the secrecy of the algorithm, but on the mathematical complexity of such algorithm. Thus, security of the watermark should not depend on the secrecy of the watermark insertion or extraction algorithm but on some system properties.

Strong Authorship Proof The watermarking scheme should be capable of inserting enough data for identification of the IP owner. The data should be of sufficient creditability to be considered as evidence in front of court for proving the authorship.

Low Embedding Cost The embedding of the watermark should be made transparent to existing design processes. The embedding cost, including both the computational cost and time needed for embedding the watermark, should be kept low.

High Reliability The reliability of a watermarking scheme can be evaluated with the robustness and probability of coincidence (P_c) of the watermark. The robustness measures the strength of the hidden signature against various attacks, while the probability of coincidence, sometimes called the false positive rate, is the possibility that the watermark is detected in a non-watermarked design. For non-repudiation, the probability of coincidence should be at least as low as the odd of finding a match fingerprint from two persons in forensic science.

Low Implementation Overhead It is usually inevitable to introduce additional overhead to the IP core after watermarking. The performance overhead, usually measured in terms of area, power and delay, should be kept low for the IP core to remain useful.

Ease of Detection and Tracking Tracking and detection is as important as watermark insertion. It is advantageous to ease the detection of watermark and enable the origin of fraudulence to be traced after possible attacks.

There may be further considerations in designing and evaluating a watermarking scheme. For example, the fairness of the scheme proposed in [7, 30]. A watermarking scheme is fair if it is able to generate watermarked designs at almost the same embedding cost and the same implementation overhead under different authorship signatures. Nonetheless, such a requirement is implicitly implied in the above seven attributes. If a watermarking scheme is not fair, the scheme will have difficulty to find a watermarked solution of high quality with an acceptable cost and effort for an arbitrary signature.

Among the seven attributes, the first six are requirements for the watermark embedding process. This does not mean that watermark extraction is an easy problem. A watermarking scheme is incomplete without a properly designed watermark extraction process. A convenient watermark detection and verification can make a watermarking scheme more practical and receptive. An unmindful watermark detection method could also weaken the robustness of the watermarking scheme.

2.3 Attack Analysis for IP Watermarking

Generally, there are three main types of attacks, i.e., *removal*, *masking* and *forging* attacks. The shared prerequisite of these attacks is that they should not degrade the design quality. That is, an obviously deteriorated design is not what an attacker wants to steal. For removal attacks, the adversary tries to eliminate the watermark completely. This task is usually very hard to succeed with the prerequisite mentioned above. Hence, the attacker may turn to tampering with the watermarked design so that the existence of the watermark cannot be detected, i.e., the masking attack. Depending on the detection mechanism, the minimum percentage of the watermark bits to be altered to result in a successful masking attack varies. A *probability of masking* (P_m) is defined as the probability that an attack would change or delete enough information to render the watermark undetectable without unacceptably deteriorating the performance of the target design [26].

In the forging attack, the adversary embeds his own watermark in the watermarked IP to claim his ownership to the design. To insert the watermark, the attacker may repeat the watermark insertion using his own signature or simply perform a ghost search. A ghost search is an attempt to make up an apparently genuine but different watermark based on the detection method of the targeted watermarked design and use it as the adversary's signature. The probability of a successful ghost search is equal to the *probability of coincidence* mentioned in Sect. 2.2.

Security analysis and countermeasures of an IP watermarking scheme against the removal and masking attacks depend on the employed mechanism for watermark insertion and detection and varies case by case. Nonetheless, there are some common analyses and countermeasures to deal with the forging attack. First, if the watermarked design is forged by merely the addition of watermark, the IP owner is able to provide an IP core with only his watermark while the attacker have only the IP core with both watermarks. It becomes obvious that the IP core belongs to the IP owner. Second, if the attacker has successfully removed or masked the IP owner's watermark and inserted his own, the time of watermark insertion becomes an evidence to distinguish which party is the legal owner. As proposed by Abdel-Hamid et al. [26], a time-stamped authenticated signature can be used. A trusted third party is engaged to act as a watermark certification authority. It is responsible for generating and distributing time-stamped signatures, as well as keeping a record for vouching the authenticity of such signatures for watermark verification in authorship proof. The dispute can be easily resolved by the time stamps of the recovered signatures.

3 Watermark Insertion

In this section, the main methods for watermark insertion in existing hardware IP watermarking schemes are presented. The detection of watermark will be discussed in Sect. 4.

3.1 Additional Functionality

An intuitive way to insert the watermark is by adding circuitry to generate the watermark. Fan et al. [31–33] proposed to embed a watermark generating circuit (WGC) into an IP core as a part of the test circuit at the behavioral design level. As shown in Fig. 3, the WGC is composed of some inverters and parallel-input-serial-output (PISO) registers. Based on the test mode signal and inverters, the watermark bits are generated in parallel. These parallel bits are then converted to a serial sequence by the PISO and stored in the shift register. Finally, the watermark sequence is combined with the test patterns in some controllable way by an arbitrator and made detectable in the test output sequence. Depending on the number of output pins used for the output test patterns, several small WGC's may be used to help reducing the overhead of watermarking.

As the WGC and test circuit is inserted to the IP core at the behavioral level and is synthesized with the functional logic, the scheme provides good protection to the IP core at all the lower design levels such as the structural and physical levels. Nonetheless, it is not a good candidate for protecting the soft IP core as the WGC can be easily distinguished at the behavioral level. By simply modifying the number of parallel bits in the WGC, i.e., the watermark length, the strength of authorship

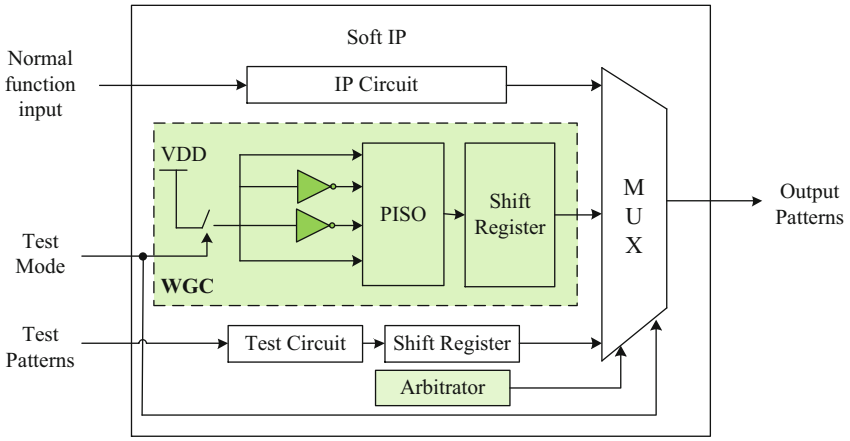

Fig. 3 Architecture of the IP core, watermark generating circuit and test circuit

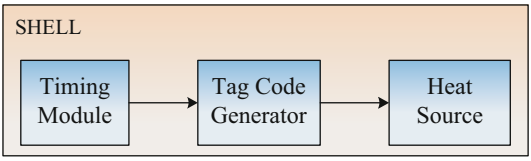

Fig. 4 The tag circuit proposed in [34]

proof can be readily controlled by this scheme. On the other hand, for a fixed WGC, the larger the IP core to be protected, the smaller the overhead due to the WGC.

Kean et al. [34] proposed to insert a small tag circuit, as shown in Fig. 4, into the target design to be protected. The heat source generates heat according to a unique tag signature (i.e., watermark) produced by the code generator, with the timing information from the timing module. The heat source is implemented with a number of parallel ring oscillators. It operates at a high frequency and generates heat when the watermark bit is '1' and turns off at the watermark bit of '0'. As a result, the tag circuit in the manufactured chip containing the IP core will generate a sequence of chip temperature changes as the unique tag signature. By measuring the chip temperature data with a thermocouple attached to the chip package and verifying the correlation between the data and tag codes in a database, the tag signature of the IP core can be identified. This is currently the only commercially available tagging approach. To enhance the security, the *Shell* around the tag circuit is fortified by some anti-tamper and reverse engineering countermeasures.

The drawback of measuring the changes in the chip package temperature is the slow data rate due to the physical limitations on how fast a chip package can heat up and how quickly a temperate measurement can be made. Instead of generating temperature changes according to the watermark, Ziener and Teich [35] proposed to convert the watermark to specific power patterns using a power pattern generator. The power pattern generator can be implemented with a set of shift registers and is controlled according to the encoding of the watermark to be transmitted. The power pattern is detected in the reset state of the IP core to avoid interference from the operational logic in the measured power. As some FPGA architectures allow the use of lookup tables (LUTs) as shift register, for FPGA implementations the authors propose to employ some functional LUTs for the shift registers of the power pattern generator. The corresponding watermark embedding process consists of two steps. First, the control logic that is responsible for emitting the authorship signature is merged with the IP core at the HDL level. Second, after logic synthesis, suitable LUTs in the functional logic are selected to implement the shift registers which are attached to the control logic in step 1. Due to the sharing of LUTs, it becomes harder for an attacker to remove or tamper with the shift registers without altering the IP functionality.

Another power based watermarking scheme is proposed by Becker et al. [36]. The tag circuit is also implemented in a high-level description. Instead of trying to detect the power signature directly, the watermark is detected based on correlation as in [34]. The scheme is superior than [35] in that it allows the watermark signal to be hidden below the noise floor of the power side-channel. This makes the watermark

detectable even when the IP core is in operation. An additional merit of hiding the watermark below the noise floor is that the watermark is hidden from third parties, making the watermark stealthier. This scheme has the same applicability as all the schemes mentioned above that it is well suited to protect the netlist and designs at lower levels than the HDL IP cores. In the high-level description, it is easy to identify and remove the tag circuit.

3.2 Additional Constraints

In each phase of IC design flow (i.e., the behavioral synthesis, logic synthesis and physical synthesis), there exist a number of NP-hard optimization problems. These problems are too complex to be solved for the exact optimum solutions requiring exhaustive enumeration. Quasi-optimal or near-optimal solutions are sought by heuristic algorithms with some design constraints. This is where constraint-based IP watermarking techniques come into play. The heuristic algorithm takes the design specifications and its performance constraints as inputs for design space exploration to select a good solution as the original IP core from a large solution space. To create a watermarked IP, the encrypted authorship message is first converted into a set of stego constraints. These constraints are then used as either additional inputs to the optimizer (i.e., pre-processing) or imposed on the output of the optimizer (i.e., post-processing). The final result will be a watermarked IP core which satisfies both the original and the stego constraints.

A generic representation of the pre-processing based watermarking procedure is shown in Fig. 5. The watermark is derived from the authorship information based on some encryption processes as depicted in Fig. 2. It is then converted into stego constraints by the constraint generator that directs the mapping from the watermark to the constraints. With the stego constrains added to the inputs of the heuristic solver, the solution space of the *original problem* is reduced to a much smaller solution space of the *stego problem*.

As the synthesis problem for generating the watermarked IP is non-linear and complex, the watermark inserted using the pre-processing approach is usually very robust. This is true especially when it is compared with the watermark inserted in the post-processing approach where the original IP core is first obtained from the synthesis problem and then altered based on the stego constraints. Such alteration may also be exploited by an attacker to mask or remove the watermark. On the other hand, synthesizing the original problem with the stego constraints is more likely to result in unpredictable design overhead. The quality of the watermarked IP cannot be guaranteed even for the optimization intensive watermarking techniques [10]. Hence, the stego constraints in the pre-processing approach must be prudently selected. This problem is not so severe in the post-processing approach. As the stego constraints are imposed on the already optimized solution to the original problem, the overhead due to the inserted watermark can be better controlled.

To limit the overhead and achieve high robustness simultaneously, a three-phase watermarking approach is proposed by Yuan et al. [37]. In the first phase, an optimized solution is obtained from the original synthesis problem. The optimized design is used as a reference in the second phase where some stego constraints are selected by considering the overall tradeoff of solution quality, watermark robustness, etc. In the third phase, the watermarked design is re-synthesized to increase the difficulty for an adversary to tamper with the watermark. As suggested by the authors, the watermark should be inserted close to the end of the synthesis to reduce the complexity of the re-synthesis process in the third phase and to increase the predictability of the change in solution quality due to watermarking.

For all the approaches described above, the strength of the authorship proof depends on the ratio of the stego problem solution space to that of the original problem. The smaller the ratio, the lower the probability of coincidence (because a solution generated under only the original constraints will be less likely to also satisfy the stego constraints), and the stronger the proof of the watermark existence.

The three generic watermark embedding procedures can be applied to various optimization problems in the SoC design process. Among them, the pre-processing approach has the widest appearance. Since its first introduction in [1, 12, 38, 39], an extensive number of constraint-based IP watermarking techniques have been proposed at different abstraction levels, which range from the system synthesis level [1, 6, 12, 13, 40, 41] to the behavioral synthesis level [42–44], logic synthesis level [44–48] and physical synthesis level [12, 44, 49, 50]. In what follows, some of the influential proposals at each level are reviewed.

3.2.1 Constraint-Based IP Watermarking at System Synthesis Level

The system to be designed by the system integrator can be treated as a large monolithic IP core to be protected by inserting a watermark during system synthesis tasks such as multiprocessor DSP code partitioning and cache line coloring.

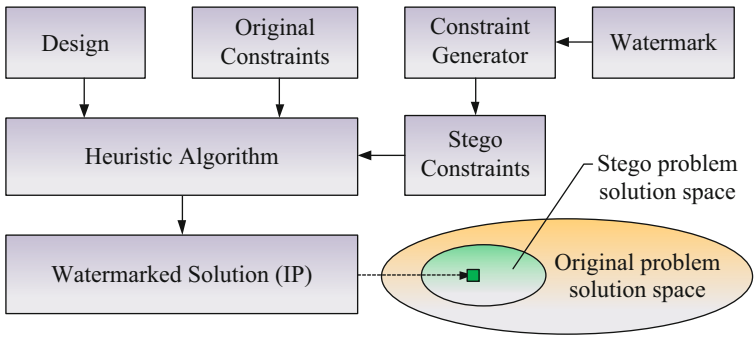

Fig. 5 A generic pre-processing constraint-based IP watermarking procedure

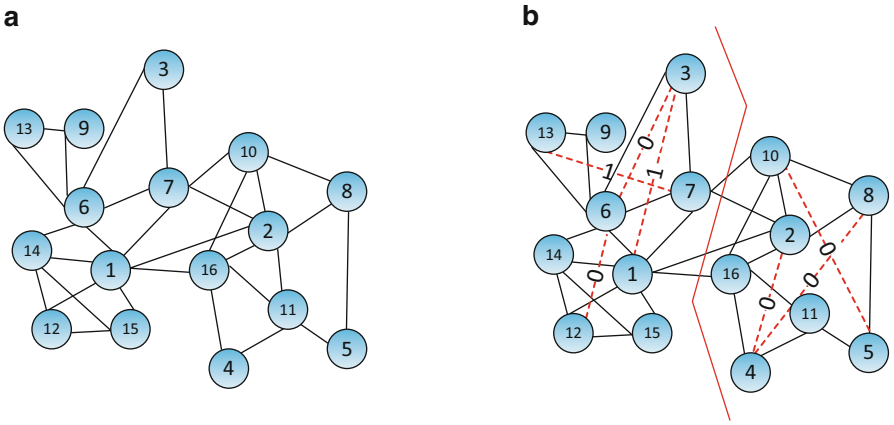

Fig. 6 An example of watermark embedding in the graph partitioning task [1]. (**a**) The graph to be partitioned with nodes indexed by integer numbers. (**b**) The partitioned graph with watermark

Watermarking based on the former task is demonstrated by the example in [1], where the watermark insertion task is formulated as a graph partitioning problem.

In the graph to be partitioned, the nodes are randomly numbered with integers, as shown in Fig. 6a. The stego constraints corresponding to the watermark mandate particular pairs of nodes to remain in the same partition. The following watermark embedding process can be used. For each watermark bit, one origin node and one terminal node are selected for pairing. The original node is selected from those nodes that have not been set as the origin node. The node with the smallest index will be chosen. The determination of the terminal node depends on the value of the watermark bit. When the watermark bit is '1', the node with the smallest odd index that has not been used in the previous pairs will be selected as the terminal node; when the watermark bit is '0', the terminal node will be the node with the smallest even index that has not been paired. Assume the letter 'A' with the ASCII code of "1000001" is a watermark (or a part of the watermark) to be embedded. Based on the embedding criteria described above, the selected pairs of nodes that need to be in the same partition are (1, 3), (2, 4), (3, 6), (4, 8), (5, 10), (6, 12) and (7, 13), each of which is connected with a dotted red line in Fig. 6b. If a balanced partitioning requires the difference in the number of nodes between two partitions to be less than 20 %, the partitioning depicted in Fig. 6b can be obtained, which represents one watermarked solution.

The watermarking scheme described above can be extended to the graph coloring problem. Graph coloring assigns labels, namely "colors", to elements of a graph subject to certain constraints. As a typical form of graph coloring, the aim is to find a way to color the vertices of a graph such that no two adjacent vertices (i.e., connected by an edge) share the same color. Many optimization problems in the VLSI design flow have been modeled as graph coloring problems. For instance, the cache-line code optimization can be solved by finding a solution to

the graph coloring problem using a given fixed number of colors, where each color corresponds to one cache line. To insert a watermark into the solution, additional edges corresponding to the watermark can be added into the graph. The watermarked solution will be generated by coloring the new graph using the minimum number of colors.

3.2.2 Constraint-Based IP Watermarking at Behavioral Synthesis Level

In behavioral synthesis, the behavioral model of the design is transformed to a RTL description to implement the behavior. The optimization tasks in behavioral synthesis, such as scheduling, assignment, allocation, transformations and template mapping, are all excellent NP-hard problems for embedding the watermark. The watermarking scheme in [43], which inserts the watermark during register allocation, is used as the example here.

After scheduling the operations with for example, a scheduled control data flow graph (CDFG), the variable values that are generated in one control step and consumed in the later steps must be stored in registers. The interval between the first time a variable is generated and the last time it is used is referred to as its lifetime. Variables whose lifetimes are not overlapped can share the same register, based on which an interval graph can be constructed. In the interval graph, each vertex denotes a variable and the edge between two vertices indicates that there is an overlap in the lifetimes of the two variables. Register allocation can then be performed by solving the graph coloring problem for the interval graph. To insert the watermark, the same approach mentioned in Sect. 3.2.1 can be used. That is, extra edges which represent the stego constraints induced by the watermark (or *watermark constraints* for short) are inserted into the interval graph. Due to the added edges, the resultant watermarked solution satisfies the watermark constraints by storing some specific variables in different registers.

Similar watermark insertion method can also be applied to other behavioral synthesis tasks. The generic watermarking approach is depicted in Fig. 7.

3.2.3 Constraint-Based IP Watermarking at Logic Synthesis Level

Logic synthesis transforms an abstract form of the design behavior (typically in the RTL HDL) to a specific design implementation constituted by logic gates. Combinational logic synthesis consists of two main optimization tasks, i.e., multi-level logic minimization and technology mapping. Both tasks can be watermarked using the scheme proposed in [45, 46]. We use the example in [46] to illustrate the watermark embedding process for a technology mapping solutions.

Given a cell library, technology mapping (also known as cell-library binding) aims to map the logic network of the design to as few library cells as possible. The complexity of finding an area-optimal solution to the problem is NP-hard [51]. Assume that a cell library consists of two cells as shown in Fig. 8a and the

logic network to be mapped is constituted by 11 gates. By performing exhaustive enumeration, we can obtain 49 optimal solutions, each of which uses six cells. The watermark can be embedded by the following procedure.

Firstly, each node (the output of a gate) which is not a primary output is uniquely identified. The set of node identifiers is denoted as $N = \{1, \ldots, 10\}$, as shown in Fig. 8b. The watermark to be embedded is a set of distinct numbers, denoted as W. The cardinality of $|W|$ is required to be smaller than that of $|N|$. Then, the watermarking constraints are imposed by specifying the nodes with identifiers equivalent to numbers in W as pseudo-primary outputs. Due to the watermarking constraints, a specific set of internal nodes will be visible in the resultant solution. If $W = \{3, 4\}$, then node 3 and 4 will be specified as pseudo-primary outputs, as depicted in Fig. 8c. The constrained network can still be solved using only six library cells. However, the number of possible solutions is reduced to 4. This means that the probability of coincidence is $P_c = 4/49$, which represents the strength of the authorship proof. For a real-life design with tens of thousands and more internal nodes, the scheme can achieve very strong proof of authorship with little or no sacrifice on the solution quality.

An alternative scheme that exploits technology mapping for watermark insertion is proposed in [52]. The watermark is inserted by breaking each selected signal node into a pair of primary output and input signals. After technology mapping and a post-processing step where the pairs of added primary output and input signals are shorted, a watermarked technology mapping solution will be generated. By applying the watermarking constraints only to those signal nodes that are not on the critical path, all the critical paths timings are preserved. Again, from the experimental results, the area overhead due to the watermark insertion is very small.

The watermarking scheme depicted in Fig. 8 can be extended to the task of multi-level logic minimization. As proposed in [45, 46], after selected nodes of the logic network are specified as pseudo-primary according to the watermarking constraints,

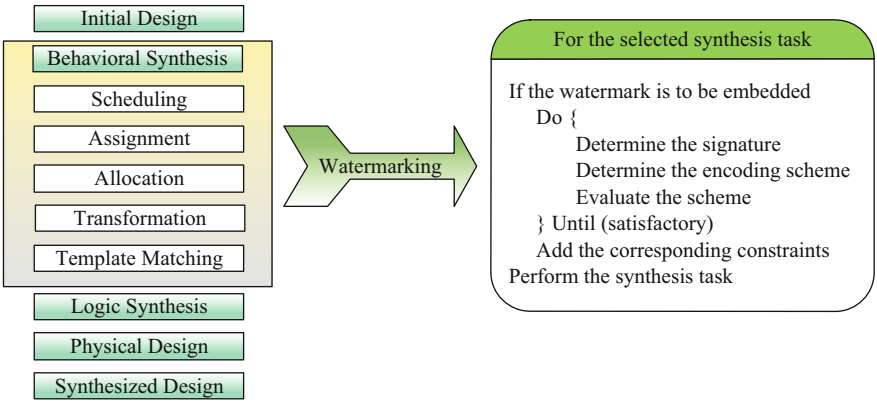

Fig. 7 A generic approach for watermark insertion in behavioral synthesis tasks [43]

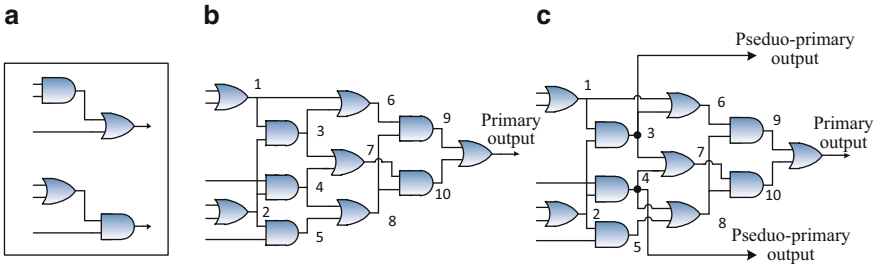

Fig. 8 An example of watermarking technology mapping solutions [45]. (**a**) Cell library. (**b**) A logic network. (**c**) The constrained logic network

an additional logic network will be augmented. The additional logic network takes the pseudo-primary output variables as its input variables and is created according to the copyright-specific pseudo-random bit-stream. The watermarked solution is conceived under the influence of the additional logic network which is to be removed in the synthesis step. The additional logic network has a very significant effect on the sub-function selection. It is almost impossible to produce the same set of sub-functions by an off-the-shelf synthesis tool without knowledge of the augmented copyright-specific logic network, which forms the basis for the authorship proof.

As an additional example, rewiring is an important synthesis technique in logic minimization. It is used for tasks such as logic optimization, delay optimization, elimination of wires with high capacitive loads or switching, etc. A redundancy addition/removal (RAR) based rewiring scheme is proposed in [47]. The basic idea is that a redundant connection may be added to some nodes in the netlist determined by the author's signature without changing the design functionality. A set of redundant connections may be generated and can be removed. The removal of the set of redundant connections makes the added connection irredundant. As a result, the added connections act as the watermark.

3.2.4 Constraint-Based IP Watermarking at Physical Synthesis Level

Physical synthesis begins with a mapped gate-level netlist generated by logic synthesis and outputs a new optimized netlist and the corresponding circuit layout. Common steps for physical synthesis include floorplanning, placement, clock tree synthesis, scan chain generation, routing, etc. Several watermarking schemes based on these tasks are proposed in [12], which are described below.

Watermarking on Path-Timing Constraints

According to the authorship signature, a set of path timing constraints are selected. To insert the watermark, these constraints are replaced with "sub-path" timing constraints. For example, the timing constraint $t(C_1 \rightarrow C_2 \cdots \rightarrow C_{10}) \leq 50$ ns of a path can be replaced with the constraints $t(C_1 \rightarrow C_2 \cdots \rightarrow C_5) \leq 20$ ns and $t(C_5 \rightarrow C_7 \cdots \rightarrow C_{10}) \leq 30$ ns of two sub-paths, where C_i denotes the i^{th} gate

in the path. The synthesis solution under the original timing constraint does not necessarily satisfy the two sub-path constraints. According to [12], the chance that satisfying the original constraint happens to satisfy both sub-path constraints is at most one-half. By constraining tens or even hundreds of timing paths in a similar way, a strong proof of authorship will be achieved.

Watermarking on Row-Based Placement
The watermark can also be inserted by constraining the placement of selected logic cells in rows with the specified row parity. That is, some cells are constrained to be placed in an even-index row while some other cells are constrained to be placed in an odd-index row. As a typical placement instance has tens of thousands of standard cells, this approach is capable of inserting a long watermark into the design and providing a high authorship proof. Nonetheless, due to the watermarking constraints, the routing cost between some cells may be increased. Hence, the logic cells that are watermarked needs to be carefully selected.

Watermarking on Standard-Cell Routing
In this approach, the watermarking constraints are the (per-net) costing of the underlying routing resource. That is, if the watermark bit is '1', unusual costs are imposed on "wrong-way" and/or via resources for the selected net, and vice versa. As a result, the watermark can be verified by checking whether a specific set of nets are unusual in their utilization of resources.

3.3 Localized and Hierarchical Watermarking

3.3.1 Localized Watermarking

All the constraint-based watermarking techniques described in Sect. 3.2 are typically realized by encoding the authorship information as a set of stego constraints, augmenting the stego constraints to the original constraints and then optimizing the design from a higher level using an off-the-shelf design tool that finally generates the watermarked IP core. These watermarking techniques employ the pre-processing approach and rely on the generation of a global solution to a design optimization problem according to a specific authorship signature. In these pre-processing based watermarking schemes, the main consideration is usually the watermark robustness and they often gloss over the process of watermark verification. As global optimization is performed for the watermark insertion, the embedded information must be recovered from the entire watermarked IP core indivisibly. A small design alteration by an attacker may cause a substantial distortion in the recovered watermark and result in the failure of watermark detection. The detection difficulty is exacerbated after the protected IP core is integrated into the SoC. In this case, to detect the watermark, the whole IP core needs to be accurately extracted from the system, which is usually very difficult and expensive even with the trapdoor of the IP placement in the SoC.

In view of these problems, Kirovski and Potkonjak [42] proposed to insert multiple *local watermarks* in the IP core to be protected. Instead of creating a large watermark, the authorship signature is converted into a set of smaller watermarks, each of which is randomly augmented into a part of the design and can be verified in its locality independently from the remainder of the design. Two behavioral synthesis tasks are exploited for inserting such watermarks, i.e., operation scheduling and template matching. Basically, both tasks consist of domains with different localities. The small watermarks are then converted into sets of additional constraints and assigned to pseudo-randomly selected domains according to the authorship signature. The scheme enables parts of the IP core to be independently protected because only a segment of the design is needed to decode the stego constraints due to the local watermark in that design segment. Moreover, the local watermarks that are independent from each other force an attacker to alter a substantial part of the IP core in order to obliterate the copyright protection.

Cui et al. [48] proposed a scheme that possesses a similar feature as localized watermarking. They insert the watermark in the technology mapping task of logic synthesis. Unlike the schemes described in Sect. 3.2.3 which require technology mapping to be performed for the whole design to generate the watermarked solution, incremental technology mapping technique is employed to synthesize part of the design for watermark insertion. In particular, a globally optimized technology mapped solution is used as a *master design*. From the master design, the slack sustainability (which determines how well a disjoint closed cone sustains its slack by replacing some of its cells in technology mapping) of disjoint closed cones are estimated. Closed clones with the product of slack and slack sustainability greater than a threshold value are qualified for hosting the watermark bits. Qualified disjoint closed cones are randomly selected and ordered by the authorship signature. The watermark is inserted into these selected closed cones by remapping them according to the constraints imposed by the signature. If the watermark bit to be inserted is '1', the selected closed cone is remapped by coercing the change of one template of the cone; otherwise, the selected template used in the cone is preserved. The watermarked solution is generated by remapping only the selected closed cones according to the stego constraints with incremental synthesis. In fact, the scheme employs the generic three phase watermarking approach described in Sect. 3.2. As the closed cones are qualified based on both slack and slack sustainability over the already optimized master design, it maximizes the embedding capacity and is stealthier than hosting the watermark bits in non-critical paths determined merely by the absolute timing slack. The timing overhead of the watermarked solution can also be kept minimal or even improved.

3.3.2 Hierarchical Watermark

Localized watermarking can be viewed as inserting multiple watermarks in the design at one abstraction level. Instead, Charbon and Torunglu [53, 54] proposed to insert multiple watermarks in different levels, laying the foundation for hierarchical

watermarking. With multiple watermarks independently embedded in different abstraction levels, a more robust protection to the IP core is provided. Only when an attacker is able to delete all the watermarks in different design level can he remove the authorship proof. One concern of the hierarchical scheme is the proportionally increase in the watermarking overhead with the number of watermarked levels. This issue can be addressed by a similar idea as localized watermarking in Sect. 3.3.1. The authorship information can be divided into small parts, and independently inserted into every abstraction level to be marked. Besides, as watermarking approaches at different levels have their own strengths and limitations, hierarchical watermarking provides a platform for these techniques to complement each other so that each technique enjoys a greater flexibility and trade-off to achieve a more robust overall IP protection scheme.

The advantage of hierarchical watermarking is exemplified by the watermarking scheme in [55], where the watermark is inserted in both the finite state machine (FSM) and the test scan chain. The FSM watermarking and test architecture watermarking are to be discussed in Sect. 4. The former has good robustness against attacks but the watermark cannot be directly detected after the IP core is integrated into the system. The latter one is vulnerable to attacks but watermark detection is very easy. Thus, the authenticity of the encapsulated FSM IP core can be conveniently detected in the field through the watermark embedded in the scan architecture while the vulnerability of the latter is fortified by the robustness of the FSM watermark. The watermark in the test structure acts like a fragile watermark. Its removal alerts the IP owner to verify the existence of the watermark in the FSM and helps to trace the pirated chips that contain the misappropriated IP core.

4 Watermark Extraction

As discussed in Sect. 2.2, a watermarking scheme is incomplete and turns out to be impractical if it is difficult to detect and track. Except the side-channel watermarking approaches, e.g., the three schemes described in Sect. 3.1, a majority of the watermarking schemes require processing the watermarked solution in order to extract the watermark.

4.1 By Physical Processing

From the watermark extraction point of view, these watermarking schemes can be classified into two types. The first type is the static watermarking scheme, whereby the presence of watermark is verified indirectly by checking if the watermarked constraints generated by the author signature are satisfied. The second type is the dynamic watermarking. For this type of schemes, the watermark can be detected from the output response by injecting a specific input sequence.

4.1.1 Static Detection of Watermark

All the constraint-based watermarking schemes described in Sect. 3.2 are static. To verify the existence of the watermark, the circuit under test (CUT) usually needs to be reverse engineered to the level where the watermarked solution is generated. Then existence of the watermark is proved by checking if the stego constraints due to the authorship signature are satisfied. There are some concerns about such a detection process. First, reverse engineering a CUT to the level where the watermarked solution was originally generated is often a costly task. Second, the verification process usually needs to expose the grammar used for generating the stego constraints from the authorship information. This can potentially weaken the security of other designs that are similarly watermarked.

Some attempts have been made to mitigate the above problems. For example, the localized watermarking described in Sect. 3.3.1 does not require the complete IP core for watermark extraction. Thus the effort and cost of retrieving the watermark will be reduced. Besides, due to the existence of multiple small watermarks in one IP core, even though a subset of the watermarks are exposed in the process of ownership proof, the remaining watermarks are still kept secret. In fact, the watermark verification process in [48] does not even leak the grammar. The process is analogous to the watermark retrieval of non-oblivious image watermarking. Since incremental technology mapping preserves the functionality of the interface ports of remapped cones, the logic functions of these nets can be retrieved from the fan-in and fan-out nodes of closed cones in the master design. To recover a watermarked cone from a marked design, corresponding nodes with the same logic functions are identified. A watermark bit of "0" or "1" can then be determined according to whether the designated template are absent or present in the cone. This watermark retrieval method possesses some degree of fragility that enables the detection of maliciously corrupted watermark bits. When equivalent fan-in and fan-out nodes of a watermarked cone cannot be found, it implies that either cells within the cone or cells surrounding the cone have been modified. Hence, the authorship can be proved by either a perfect or high match between the recovered bit stream and the embedded watermark. Throughout the verification process, the grammar used to generate the watermarking constraints is not exposed.

As an effort to avoid leaking the grammar used to generate the watermarking constraints, Qu [56] proposed a public watermarking approach, where the watermark is divided into a public part and a private part. The private part of the watermark is embedded in a secret way as in the traditional constraint-based watermarking while the public part is embedded in designated locations by methods made known to the public to allow public detectability. This way, detection of the watermark is made easier than that of the conventional schemes but the secret watermark is much well protected. Only when the public part of the watermark is suspected to be attacked will its private portion be recovered for authorship proof. Essentially, this scheme can be deemed as a variant of localized watermarking.

4.1.2 Dynamic Detection of Watermark

Unlike static watermarking schemes, dynamic watermarking does not require the watermarked design to be reverse engineered to the level where the watermark is inserted to perform the watermark detection. Instead, dynamic schemes are characterized by the watermark stimuli-response pair whereby the watermark bits can be detected from the output response by running the watermarked design with a specific input sequence. Such an idea appeared first in [1], where it was proposed to embed a covert channel into a design in such a way that only the authors of a design can observe and interpret information obtained through the channel. The authors illustrate the idea with an example of a digital bandpass filter. The stego constraints which encode a designer signature are added in the filter structure and by observing the outputs for the specific input segments, the embedded message can be identified.

For dynamic watermarking schemes, state transition graphs (STG) of finite state machine (FSM) at behavioral level and test structure such as scan chains at design-for-testability (DfT) level are the two common vehicles. Typical schemes based on these two vehicles are discussed below.

FSM Watermarking

The first FSM watermarking scheme was proposed by Torunoglu and Charbon [57]. This scheme starts with building the FSM representation of the sequential design. Then, unused transitions in the STG are extracted and the watermark is inserted by adding correspondingly defined input/output sequences. In case of completely specified finite state machine (CSFSM), an auxiliary input variable is added to expand the FSM. To satisfy the required strength of authorship proof, the minimum number of transitions needed is calculated and compared with the maximum number of free transitions. If the probability that a non-watermarked design carrying the watermark by coincidence is not low enough, more auxiliary inputs are added. The input sequence is randomized with the set of unused transition inputs to produce an output response that contains the encrypted authorship information. To read the watermark, one should have both the input sequence and the secret key. This scheme works at a high level of the design flow, which provides extra strength. It enables the embedded watermark to be detected dynamically at almost all lower levels, even after design manufacturing. At the same time, the watermark is immune to FSM reduction techniques and is very hard to remove, as the variables used are usually part of other transitions. Nonetheless, this scheme has two deficiencies. Firstly, finding an input sequence that satisfies the required low probability of coincidence and does not incur a high overhead on the STG is an NP-hard problem [15]. Although the authors proposed to use exhaustive search or Monte-Carlo search [58] as the solution, the overhead incurred in the design phase is still high. Secondly, the scheme just makes monotonous use of the unspecified transitions of the STG for watermark insertion. The embedding capacity is limited by the number of free input combinations. If the watermark length is increased beyond the available number of unspecified transitions to boost the authorship proof, the overhead aggravates rapidly.

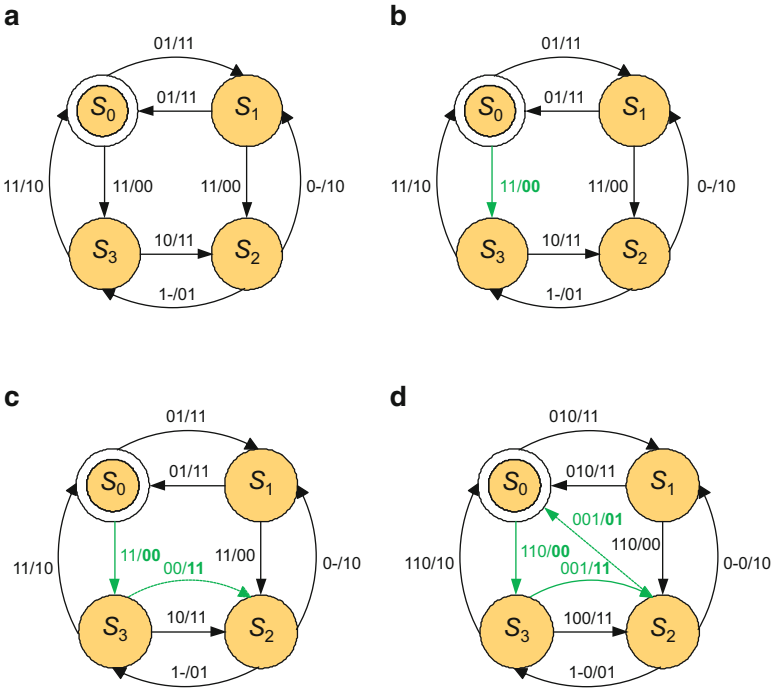

Fig. 9 An example for the watermarking scheme proposed in [59]. (**a**) The original STG. (**b**) The output "00" of S_0 coincides with the first watermark segment. (**c**) One transition added for S_3 using its non-specified input to coincide with the second watermark segment. (**d**) One transition added for S_2 by adding one input bit to coincide with the second watermark segment

To overcome the weaknesses of the previous scheme, Abdel-Hamid et al. [59] proposed to utilize existing transitions as well as unspecified ones in an output mapping algorithm to watermark the FSM, which helps to increase the watermark embedding capacity and lower the overhead. For example, assume a watermark of "001101" is to be inserted into the STG in Fig. 9a. As each transition output consists of two bits, the watermark is partitioned into three segments, i.e., "00"–"11"–"01". Starting from the stage S_0, a coinciding output "00" is found with the next stage being S_3 (see Fig. 9b). The state S_3 has no output being "11" but it has unspecified inputs. A transition (00/11) from S_3 to S_2 is added into the STG, which yields the STG shown in Fig. 9c. In S_2, there is no output coinciding with the third segment, i.e., "01", of the watermark, and all inputs are specified. In this case, an extra input bit has to be added to expand the STG. The extra bit is assigned '0' for all existing transitions in the original STG, and '1' for the added transitions. Therefore, a transition (001/01) from S_2 to S_0 is added as the third watermarked transition, as shown in Fig. 9d. When the input sequence "110001001" is injected into the watermarked FSM at state S_0, the watermark can be retrieved from its output sequence.

The embedding process is fast as the watermark bits are inserted at large by a random walk of the STG. When all output bits of an existing transition of a visited node coincide with a substring of the watermark, that transition is employed as part of the watermarking sequence. Otherwise, one of the unspecified transitions is selected to insert some watermark bits. One drawback of this scheme is that coincidence of existing transition's output with the watermark bits becomes rare when the FSM has a high number of output bits. When only unspecified transitions are used for watermarking, this scheme becomes similar to the previous scheme. If there is not enough unspecified transition, pseudo input variables have to be added. Another drawback is the fixed assignment made to the pseudo input variables. This conspicuous discrimination between existing transitions and added transitions may unveil the pseudo-input to an attacker, and the removal of any pseudo input can easily eliminate or corrupt the watermark without affecting the FSM functionality. In addition, the addition of new input variables with fixed assignments on all transitions increases the decoder logics and hence the overhead of watermarked FSM significantly.

A more robust and lower overhead version of STG transition-based watermarking scheme was presented in [60]. To overcome the problem of low probability of coincidence between the watermark substring and existing transition outputs, a longer verification sequence is suggested which results in a response sequence with more bits. In the previous two schemes, the number of watermarked transitions is equal to m/k, where m is the watermark length and k is the number of output bits for each transition. In contrast, the number of possibly watermarked transitions is made larger than m/k in this scheme. The localities of the watermark bits are randomly generated and dispersed into the bits of the output sequence. The probability that the designated output bit has the same value as its hosted watermark bit is ½. This tremendously increases the opportunity of employing existing transitions for watermark insertion, reduces the need to add new transitions and consequently reduces the overhead of watermarking. In addition, instead of making the fixed assignment of auxiliary input variables, this scheme keeps their states as don't cares in all transitions except only for the watermarked transitions where their input combinations need to be fixed to host the watermark bits. There are two advantages for this deferred assignment. Firstly, it minimizes the output and next state decoders of the watermarked design. Secondly, the added transitions become indiscernible from the existing ones and removal of the pseudo input variables can affect the FSM functionality. The drawback is that a longer verification sequence and output sequence will be required for watermark retrieval.

The states of the STG can also be utilized for watermark insertion. For example, in [15], the FSM is watermarked by introducing redundancy in the STG so that some exclusively generated circuit properties are exhibited to uniquely identify the IP author. According to the watermark, a specific sequence of states is generated and will only be traversed with the excitation of a specific sequence of inputs. The watermark verification relies on the presence of such extraneous states in the STG. However, the watermark will not survive the removal of all redundant states by state minimization attack. The author provides two possible ways to verify the presence

of the watermark, which are the implicit Binary Decision Diagram (BDD)-based enumeration method and the Automatic Test Pattern Generation (ATPG)-based method. The former is too slow for large circuits, whereas the latter requires the solution of an NP-complete problem. It is also not evident that the verification can be carried out efficiently on large circuits.

Test Structure Watermarking

The main limitation of FSM watermarking is that once the watermarked IP is integrated into the chip, the watermarks hidden in the SOC after the chip has been packaged cannot be extracted in the field without dismantling the encapsulation. The only signal that can be traced after chip packaging is the test signal. Thus, numerous post-fabrication verifiable schemes based on the test sequence have been proposed.

In [61], Kirovski and Potkonjak developed a technique for watermarking the design at the logic network level during the selection of the chain of scan registers for sequential logic test generation. The watermark is converted into additional user specific constraints for the selection algorithm. The insertion is realized by using a set of protocols for standardized ordering of the directed graph representation of the circuit. Due to the specific nature of the design of partial scan chains, watermark detection becomes a trivial procedure which is performed by inserting a standard set of test vectors and receiving a set of outputs from the scan chain that is uniquely related to the embedded signature. Similarly, Cui and Chang [62] proposed to add the constraints generated by the owner's digital signature onto the NP-hard problem of ordering the scan cells to achieve a watermarked solution that minimizes the test power. As only the order of scan cells changes, this scheme has no impact on the fault coverage and test application time. The first scheme [33] described in Sect. 3.1 also combines a watermark generating circuit with the test circuit of the IP core at the behavior design level. The watermark sequence is observable from the response patterns in the test mode. All these schemes enable field authentication of the authorship by the IP buyer after the chip has been packaged. However, as the watermarked test core of these schemes is independent of the functional logic, it is not a difficult task for an attacker to redesign the test circuit in order to completely remove or partially corrupt the watermark.

To cope with the limitations, Chang and Cui [63] introduced a synthesis-for-testability (SfT) watermarking scheme. The watermark is embedded as implicit constraints to the scan chain ordering problem as in [62]. Unlike the DfT watermarking methods [33, 61, 62], which are performed after logic synthesis, the SfT watermarking scheme inserts the watermark into the scan chain before logic synthesis. This helps to merge the test functions with the core functions, making the attempts to remove or alter the embedded watermark more costly as such attempts are now quite likely to impact the design specification and optimality. In addition, by using the SfT technique instead of the DfT technique, standard flip-flops can be used as scan register. Without the need to use specialized scan flip-flop (SFF) cells, this scheme is applicable to the protection of IP cores in programmable logic devices. The possible concern of this scheme is that SfT technique is not the mainstream testing technique in industry.

4.2 Side Channel Watermark Extraction

Until recently, efficient watermark detection becomes a rising focus, which results in the emergence of many FSM and test structure based watermarking schemes. As another attempt to simplify the watermark detection, side-channel based watermarking is proposed, where the watermark is detected from side channels such as temperature, power, electromagnetic radiation, etc. Three representatives of such schemes [34–36] have been described in Sect. 3.1. Among them, the scheme by Kean et al. [34] provides the most convenient way to detect the watermark. A thermocouple is attached to the chip package for measuring the temperature changes and the watermark is detected by verifying the correlation between th emeasured data and a database of watermarks. The slow data rate for this temperature based scheme may be a drawback. However, as pointed out by the authors, the time consumed by the measurement is acceptable compared with the alternative of extracting the chip from a system and sending it to a lab for analysis. The measurement process is also much faster than those methods that require electrical contact, considering the time it takes to locate suitable probe-points on the circuit board and connect probes [34].

When multiple watermarked IP cores are integrated together in the system, the emitted side channel signals representing the watermarks of different IP cores will be superposed, leading to interferences among the signals and an increase in complexity to decode the watermarks. This observation holds especially for the power based watermarking scheme in [35] where the power signature is measured directly. To enable the decoding of all the power signatures, the authors proposed a multiplexing method in [64], where the communication channel is divided into multiple logical information channels and one information channel is used for a power signature. The authors explored four different multiplexing methods, i.e., space division multiplexing (e.g., different power pins), time division multiplexing (e.g., different time slots), frequency division multiplexing (e.g., different carrier frequencies), and code division multiplexing (e.g., different decoding codes). The multiplexing methods work best if there is a system-level plan on how the different power signatures in different IP cores are to be multiplexed and decoded. As the IP cores are usually developed by different IP owners, such a requirement may not be easy to satisfy.

The power based side-channel watermarking scheme in [36] is immune to the interferences among multiple power signals of the watermarks. This is because the power signature is detected based on the correlation of the power data and the tag codes stored in a database. The long tag code is pseudo-randomly generated according to the authorship signature. Each tag code is unique if it is sufficiently long. The detection mechanism enables the watermark to be hidden below the noise floor of the power profile. While ensuring that the watermark is easy to be detected in the power side channel, the scheme prevents an arbitrary third party from detecting the watermark.

5 Fingerprinting

The watermarking techniques described above, though capable of identifying the IP ownership, is unable to trace the guilty IP buyer from the unauthorized resold copies. This is because the distributed IP instances to different buyers are identical and carry the same watermark. The problem can be resolved by fingerprinting which embeds a unique and distinguishable mark (i.e., fingerprint) into each distributed IP instance. The malicious buyer who illegally redistributes his IP instance to another entity or misuses his IP instance in an unauthorized application can be easily identified based on the embedded fingerprint.

IP fingerprinting shares very similar desiderata with IP watermarking, such as functionality preservation, high credibility of the proof, low embedding cost and design overhead, high robustness against attacks, transparency to existing design flows, and ease of verification. These desired attributes have been explained in Sect. 2.2. However, the requirements of low embedding cost and high robustness against attacks are more stringent for fingerprinting. The differences are highlighted by examining these attributes in the context of fingerprinting.

Low Embedding Cost A large quantity of distinct high-quality fingerprinted IP instances, instead of the same instance, need to be generated for different buyers. Hence, the incremental embedding cost (mainly the time and effort) for each instance must be kept reasonably low. Caldwell et al. [65, 66] suggest that the run-time for creating every additional fingerprinted instance should be much less than that for solving the synthesis task from scratch. Qu and Potkonjak [8] even advocate that an effective fingerprinting scheme should be able to create $K >> 1$ fingerprinted IP instances at the expense close to that for finding one single solution.

High Robustness Against Attacks The fingerprint needs to be robust against all attacks that can remove or mask a watermark. In addition, the fingerprint must also be collusion-secure. In a fingerprinting scheme, different IP buyers receive different fingerprinted instances. If the inserted fingerprints are the only disparities in different instances, it will be relatively easy for a group of malicious buyers to collude and remove the fingerprint (or more commonly, forge an IP instance from which no buyer in the colluding group can be traced). The collusion attack is especially straightforward if the fingerprint is independent of the functional logic. Hence, Qu and Potkonjak [8] postulate that the IP instance of an innocent buyer cannot be created from any combination of distributed fingerprinted IP instances of a robust fingerprinting scheme and at least one of the guilty buyer can be traced from the forged instance created by a collusion attack. As a rule of thumb, the fingerprint should be closely coupled with the functional logic, making any attempt to remove the fingerprint without affecting the IP functionality or rendering the IP instance useless impossible; the distributed fingerprinted instances should be structurally diverse to disguise the differences incurred by different fingerprints.

It is obvious that the intuitive way to create each fingerprinted instance by repeating the entire process of a typical constraint-based watermarking technique is impractical. While the required engineering time and effort for creating a single

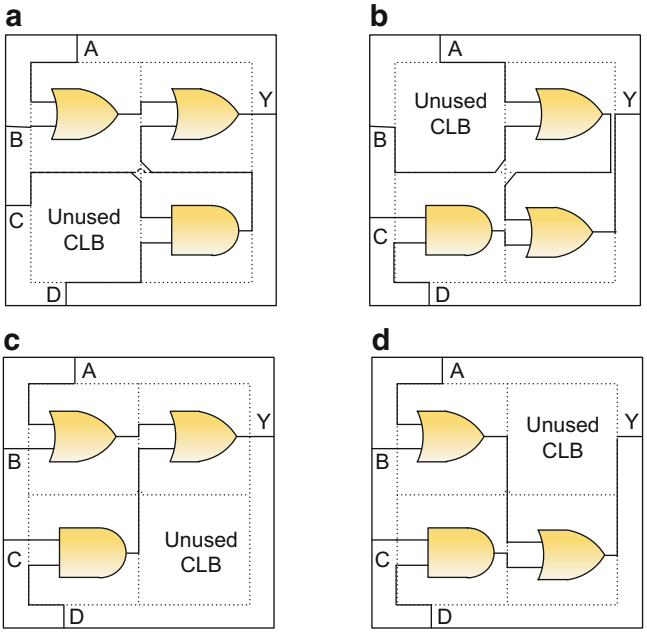

Fig. 10 Four instances (**a**)–(**d**) of the same function in a tile with fixed interface [67]

high-quality watermarked IP instance may be acceptably low, due to the multiplying factor for a large number of IP buyers, the cumulative fingerprinting cost is huge and unrealistic. In fact, only a few proposals [3, 8, 65–68] have managed to reduce the total cost to a reasonable level, which are to be illustrated below.

Lach et al. [3, 67] proposed the first IP fingerprinting scheme for FPGA designs. The FPGA design can be partitioned into small tiles and each tile has several structurally different but functionally equivalent instances. For example, consider a segment of the design for a Boolean function $Y = (A \vee B) \vee (C \wedge D)$. This Boolean function can be implemented in a tile containing four configurable logic blocks (CLB's) and there are four different implementations that are interchangeable as shown in Fig. 10a–d. For all the four implementations, there is one unutilized CLB which can be used to hold a part of the IP buyer's fingerprint (and also the IP owner's watermark). If the same implementation in Fig. 10a is used for all fingerprinted instances distributed to different IP buyers, the difference in the tile among fingerprinted instances will be ascribed only to the inserted fingerprint, and simple comparison collusion will reveal the fingerprint bits. In contrast, by randomly using one of the four implementations for the Boolean function, the difference of various IP buyers' fingerprints will be disguised by the variation in functional structures. Attempts to tamper with the differences revealed by comparing fingerprinted IP instances may alter the functionality of the design and render the design useless.

Due to the employed tiling and partitioning technique, the required effort for generating each fingerprinted instance is greatly reduced. Using the same example depicted in Fig. 10, where the FPGA design is partitioned into four tiles and each tile can be implemented with four different instances, $4^4 = 256$ instances with different functional logic structures can be obtained. Assume that the required effort to place and route an entire design is E, then the effort required for the implementation of each tile instance is around $E/4$. The effort to implement each tile instance, amortized over its use in $4^3 = 64$ different design instances is $E/(4 \times 64)$. Hence the effective effort to generate each tiled design is only $E/64$.

As the scheme employs only the unused CLB's in an FPGA design, which always exist, the area overhead is low. However, creating the tiled design and generating functionally equivalent but structurally different instances for each tile may introduce non-negligible timing overhead for the resultant design. The vulnerability of this scheme is that the fingerprint is independent of the functional logic after all. The fingerprint can be removed by reverse engineering the fingerprinted IP instance to an abstraction level before the fingerprint was inserted.

As opposed to the previous scheme which is only applicable to designs with specific regular and highly granular structure, the scheme proposed by Caldwell et al. [65, 66] can be applied to almost all synthesis problems. The basic idea is to obtain an initial seed design by performing the synthesis task from scratch. For each IP buyer, a new fingerprinted instance can be generated with the buyer's fingerprint and the seed design, where only incremental iterative optimization is required. As the effort for creating the seed design is leveraged for generating each fingerprinted design, the scheme manages to reduce the cost of fingerprinting tremendously. Take the standard-cell placement problem for example. The target of this problem is to place each cell of a gate-level netlist onto a legal site with no overlap between two cells and minimized interconnection wire lengths. To create a fingerprinted solution, an initial placement solution S_0 is obtained from scratch. Then, a subset of signal nets N' in the design is selected according to the fingerprint. The weight of each net in N' is set to 10 with the remaining nets set to 1. Based on the solution S_0 and the new net weights, the fingerprinted placement solution can be generated by incrementally replacing the design. The pseudo-code of the fingerprinting process is depicted in Fig. 11. It should be noted that the seed solution for generating the fingerprinted solution S_i may not necessarily be S_0. Instead, the solution S_{i-1}, i.e., the fingerprinted solution for the $(i–1)$-th user, can be used. As S_{i-1} is a local optimal solution, re-weighting selected signal nets will break the local optimality and facilitate the generation of a new fingerprinted solution S_i that is far away from S_{i-1}. Hence, the new seed design will help to generate fingerprinted instances that are more resilient against collusion attacks. However, as mentioned by the authors, the new solution will inherit the constraints from the previous fingerprinted solution and affect its quality.

It is worth noting that the iterative fingerprinting approach can also be applied to optimization problems that cannot be solved by iterative improvement. For example, given an SAT problem, new solutions cannot be generated from a seed solution by applying iterative improvement techniques. However, a fingerprinted solution can

1. Obtain an initial placement solution S_0 in LEF/DEF format.
2. For i = 1 to n (n is the number of IP buyers)
3. Select a subset $N' \subset N$ of the signal nets in the design according to the i-th user's fingerprint.
4. Reset the weights of all signal nets to 1.
5. Set the weight of each net in N' to 10.
6. Incrementally replace the design, based on the seed solution S_0 and new net weights, to obtain the fingerprinted placement solution S_i.

Fig. 11 Pseudo-code of the iterative fingerprinting approach on standard-cell placement

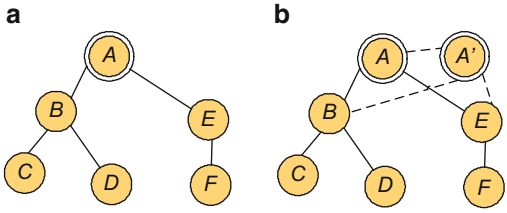

Fig. 12 Duplicating vertex A to generate a solution from which two solutions can be derived for the original graph. (**a**) Original graph (**b**) New graph

be generated by solving a new SAT problem with smaller size which is built by preserving the assignments of variables selected according to the fingerprint and removing them (along with their complements) from the initial SAT problem.

In the iterative fingerprinting approach, the incremental optimization effort required to generate each fingerprinted IP instance is still non-trivial. As a large number of fingerprinting instances are usually required, the total amount of effort may still be huge. Qu and Potkonjak [8] proposed a scheme with almost zero incremental effort to generate various fingerprinted solutions after a seed solution is found. The key idea is to introduce a set of independently relaxable constraints before solving the original design problem. From the obtained solution for the modified problem (namely the seed solution), a number of distinct solutions for the original design problem can be derived by independently relaxing each constraint. The authors proved this idea on the NP-complete graph coloring problem. Additional constraints can be introduced by duplicating a selected set of vertices, modifying small cliques or adding edges between unconnected vertices. Take for example the technique of duplicating a selected set of vertices. Assume we have an original graph shown in Fig. 12a. We duplicate vertex A by adding vertex A' and connecting A' with all the neighbors of A. An edge is also added between vertices A and A', resulting in the new graph shown in Fig. 12b. After solving the graph coloring problem for the new graph, a coloring solution can be obtained, with vertices A and A' colored differently. From this solution, two coloring solutions for the original graph can be easily derived with vertex A taking one of the two different colors. Similarly, if k vertices are duplicated and one solution is found for the new graph, a total of 2^k different solutions can be derived for the original graph.

The three techniques mentioned above modify the graph to introduce additional constraints before the graph coloring problem is solved, and hence can be classified as the pre-processing approach. The quality of the derived solutions heavily depends on the added constraints. If the original graph is overly constrained, more colors may be used for the modified graph than the necessary number of colors for the original graph. To cope with this problem, the authors also present a post processing approach. A solution to the graph G (V, E) with k colors is first obtained. Then the vertices of the graph is partitioned into m subsets by their colors with the number of colors in each subset being C_1, C_2, ..., C_m, respectively. The sub-graph formed by the i-th subset of vertices is C_i colorable. As the size of each sub-graph is in general relatively small, all the C_i-color solutions can be found by exhaustive enumeration, denoted as n_i. In this manner, all the solutions for each sub-graph can be exhaustively found and a total number of $n_1 \times n_2 \cdots \times n_m$ solutions can be derived for the graph.

The scheme, although very effective in generating a lot of different solutions with low overhead, lacks a clear mechanism to insert the fingerprint of high credibility. Although each derived solution is guaranteed to be distinct, how distinct the fingerprinted instances are remains a question. If many similarities exist among fingerprinted solutions, the solutions will be vulnerable to collusion attacks.

All the three fingerprinting schemes described above focus on how to create a large number of different quality fingerprinted instances with a low cost and how to make the inserted fingerprint robust against removal, masking and collusion attacks. As in the early-stage watermarking schemes, the ease of fingerprint verification has been neglected. The fingerprint is verified by checking whether the corresponding constraints are satisfied, which is very similar to that of the static watermarking scheme introduced in Sect. 4.1.1. This static type of fingerprint verification process is cumbersome and costly. The verification complexity escalates after the fingerprinted IP is integrated and packaged into a system.

More recently, Chang and Zhang [68] proposed a dynamic fingerprinting scheme that allows the embedded fingerprint to be conveniently detected off-chip. The fingerprint is inserted by constraining the state encoding of a test machine to be embedded into and synthesized with a sequential circuit IP. A test machine [69] is a special FSM that has the following property: An n-FF test machine can be set to any of its state with a corresponding n-bit *synchronizing sequence*; each state of the test machine can be distinguished by applying any n-bit input sequence and observing its output sequence (termed *distinguishing response*). The test machine can be used as an alternative to the scan chain for improving the controllability and observability of sequential circuit IP. As the test machine is synthesized with the design, the fingerprint encoded into the state variables is well integrated with the functional logic and has a global influence on the circuit structure, making the fingerprinted instances inherently collusion-resistant.

The detection of the fingerprint is straightforward. By injecting a specific fingerprint verification sequence, a corresponding distinguishing response can be obtained. The fingerprint bits can then be extracted from the response sequence.

As the fingerprint is detected from the test output, the verification process can be performed conveniently in the field after the fingerprinted IP is integrated in a system.

Typically, the IP owner's watermark and buyer's fingerprint is inserted by concatenation. This naive way doubles the signature length and increases the stego constraints. This issue is first addressed in [68] by using a single fused signature that carries both the watermark and the fingerprint information. As this secure fused signature can be applied to any fingerprinting scheme, its method of generation is described in details below.

The signature is generated through a blind signature protocol [70]. For simplicity, Chaum's blind signature protocol based on RSA is considered. Let (n_A, e_A) and d_A be the certified RSA public and private keys of Buyer A, where $n_A = p_A \cdot q_A$ is the product of two large random primes. The fingerprint F to be embedded into the IP instance for Buyer A can be generated through the following message exchange between the IP provider and Buyer A.

1. *Watermark generation*: The IP provider selects a message to convey its ownership information. The ASCII encoded message is first converted into a binary string and then processed by a hash function such as SHA-2 [71] to generate an integer W, where $0 \leq W < n_A$.
2. *Blinding phase*: When Buyer A makes a purchase request, the IP provider randomly selects a secret integer k_A satisfying $0 \leq k_A < n_A$ and $gcd(n_A, k_A) = 1$ to compute $W_A = W \cdot (k_A)^{e_A} \mod n_A$. This concealed watermark W_A is then sent to Buyer A.
3. *Signing phase*: To endorse the purchase, Buyer A signs W_A with his secret key d_A and returns his blinded signature $F_A = (W_A)^{d_A} \mod n_A$ to the IP provider.
4. *Un-blinding phase*: By computing $F = F_A \cdot {k_A}^{-1} \mod n_A$, the IP provider obtains the signature of Buyer A on his watermark W, which is the fingerprint to be inserted into the IP instance sold to Buyer A.

The IP provider can verify if the fingerprint F is genuine by decrypting it with Buyer A's public key. Since F is the digital signature of Buyer A on the IP provider's watermark W, it provides an undeniable proof for tracing the redistribution of the copies of IP bought by him. Meantime, as W is concealed in W_A, Buyer A has no knowledge of the IP provider's watermark W and his own signature on W. The blindness of F increases the deterrent effect of uncertainty.

6 Related IP Management and Protection Techniques

There are a large number of IP management and protection techniques that are, to a large or a small extent, related to hardware IP watermarking and fingerprinting. We briefly survey some of them and emphasize their relationship to hardware IP watermarking and fingerprinting.

ID Extraction
The cost of modern transistors is very low, but the cost of modern silicon foundries is very high, even above $1 billion. The manufacturing and testing processes are very complicated and require careful and accurate tracking of each wafer and each integrated circuit. Due to intrinsic process variation each integrated circuit is unique. It has been observed even before the first hardware watermarking and fingerprinting techniques were introduced that it is rather easy to extract a reasonably stable ID for each integrated circuit that can be used for its tracking during the manufacturing and testing processes [72]. There are three key differences between ID extraction and hardware watermarking. The first is that ID extraction is conducted in a secure environment: there is no need to consider attacks. The second is that each chip must have a unique identifier and there is no need for recognizing watermarks. Finally, the extraction of the ID is easy since the chips are still not packaged and are anyhow subject to testing and characterization.

Remote Hardware Enabling and Disabling
There are three essential steps in watermarking-based hardware IPP. By far the best addressed is watermark embedding. Also, it was recognized that watermark extraction is an important hardware IPP task. While initially it was assumed that complete reverse engineering of integrated circuits is a relatively easy task, several techniques later recognized that in many scenarios remote detection of hardware IP is beneficial [73–75]. Also, several hardware watermarking techniques automatically enable easy remote watermark extraction or even make that each output produced by the watermarked circuitry contains information about the watermark [15, 16]. The importance of the third step, watermark enforcement, was recognized more recently. The idea is that only the designer can issue commands for IC activation or deactivation. There are several variants of the basic schemes but all of them share the same IPP mechanisms. Essentially each chip has a physical unclonable function that produces outputs (key) that are required for correct operation of the chip. Only the designer can produce the key after the manufacturer or the owner of the chip provides certain IC measurements [74, 76–81]. While, of course, remote circuit enabling and disabling is more powerful than just watermark embedding as it regularly results in significantly higher hardware and sometimes even timing overheads. The final remark is that recently several techniques have emerged that enable remote software enabling and disabling using hardware-based primitives such as PUFs [82, 83].

Intentional Hardware Trojans
Hardware Trojans are widely studied malicious circuitries that are embedded by attackers in order to enable certain actions that are detrimental to legal owners of the circuits. Typical objectives include extraction of privileged information and targeted denial of service. It has been widely acknowledged that due to process variation and the ultra large scale of integration it is often very difficult to discover hardware Trojans that are embedded either during design or manufacturing of integrated circuits [84, 85]. Interestingly, one can intentionally embedded hardware Trojans in its own designs and chips in such a way that they facilitate hardware

and software IPP [85]. For example, process variation can make only a subset of potential Trojans active. The same results can be achieved if some parts of each chip are intentionally damaged or disconnected. Now, only the manufacturer of the chip knows all non-operating components on a particular chip. So, the relevant compile company may use this information in such a way that each piece of software is compiled in a unique way for each integrated circuit. Hence, software piracy is prevented. Interestingly, intentional hardware Trojans can be realized in such a way that the average and maximal timing degradation is nominal [82, 83].

Hardware Obfuscation

Hardware obfuscation is a technique that implements a piece of digital logic in such a way that it is very difficult and preferably impossible to reverse engineer its functionality. An excellent example of potentially very effective hardware obfuscation approach has been recently proposed and analyzed by researchers at NYU Polytechnic [86]. The key idea is to implement pairs of similar gates in such a way that there is a very small difference in their realization that is difficult for reliable characterization even using destructive reverse engineering techniques. In some sense it employs exactly the same argument that has been used for securing Actel/Microsemi FPGA devices that if one uses a large number of very small fuses, the device is difficult to reliably reverse engineer. The question about resiliency against side channel attacks is currently open. Some recent reverse engineering techniques employ PUFs to implement the logic. Therefore, each chip of the same design has unique structure and unless a small percentage of configurable logic is properly configured, the chip is not functional. Hence the direct manufacturing of copied designs is not possible and the attacker is forced to completely logically reverse engineer the pertinent design. This technique can be easily retargeted in such a way to ensure protection of software intellectual property as well as privacy of data against side channel attacks [82, 83, 87]. The key idea here is to make unique software for each chip of a particular design and/or to encrypt data by encrypting and decrypting it via PUFs.

Hardware Metering

An interesting and important question is how many integrated circuits are produced without permission. For example, this information can be used to figure out what is the size of the monetary damage. Another interesting and somewhat related hardware metering question is how often a particular IP core on a specific chip is used. The first question can be addressed in several ways. For example, one hardware metering technique uses the fish paradox [88] to estimate the number of chip copies [89, 90]. This basic idea can be further augmented to take properties of process variation for more accurate estimation.

Software Metering

Another interesting and economically important question is how many times a particular software program is executed on a specific integrated circuit. This question can be addressed by observing the level of device aging at a set or a system of transistors [90–92]. If there are initial measurements of threshold voltage at these

transistors one can use a simple formula to figure out the aging, i.e. how often the channel of each transistor was under pressure because it was in open switch mode. If there is no such initial measurements, one has to conduct two threshold voltage measurements, one before and one after small intentional aging mechanisms, in order to find the initial position on the aging curve [90]. Note that the same approach can be used for hardware usage metering as defined in the previous short summary.

Foundry Identification
Accurate tracing of the origin of an integrated circuit is an interesting problem with many applications. For example, identifying the source of counterfeited circuits or chips that are produced without proper authorization are economically and strategically important problems. The problem is algorithmically and statistically interesting because now one has to identify a set of statistical properties that are strongly correlated with known chips that are fabricated by a specific silicon foundry. Wendt et al. recently proposed use of Kolmogorov-Smirnov procedure for this task [87].

We finish our brief outline of hardware watermarking and fingerprinting techniques by observing that there is an interesting relationship between them and several other security tasks such as secure remote sensing, secure remote computation, and hardware and software attestation. We indicate in the next section that we believe that the basic concepts of hardware IP watermarking may be used for the creation of qualitatively new security approaches for semantic protection of design intellectual property.

7 Challenges and Opportunities

There are numerous directions that researchers of hardware IP watermarking and fingerprinting are currently pursuing. Maybe the most interesting and important observation is the rapidly growing trend of shifting the initial emphasis from embedding watermarks into design toward their extraction and IP active protection using enabling and disabling. Another interesting trend is towards quantitative evaluation of how many non-authorized ICs are produced and how much they are used.

In this section we focus our attention on three directions that have as their primary starting points on hardware watermarking and fingerprinting techniques that may impact not just hardware security and protection but also software rights enforcement and emerging areas such as big data and the Internet of Things.

Trusted IP Modules
The exponentially growing discrepancy between silicon and designer productivity implies a need for hardware and software reuse. The essential requirement for hardware and software IP is that these functional artifacts are trusted, i.e. they implement their declared functionality and nothing more. In more demanding

scenarios they also should not contain any security vulnerabilities. The creation of trusted hardware and software has attracted surprisingly little attention [93]. The basic idea of this effort is that each IP resource is used in each cycle by the declared functionality such that there exists very little resources and time in which an attacker can launch his attack. The additional focus is placed on the minimization of energy and on low time overhead. Obviously, there is large room and potential in this domain that requires new and fresh ideas and concepts.

Support for Trusted Software Execution
There are only a few actual reported and documented hardware IP attacks. At the same time new software attacks are reported on a daily basis. Only software piracy is estimated on well over $100 billion [94]. Therefore, the importance of software IP protection is of paramount importance. Numerous watermarking and other techniques have been proposed for these tasks [95–97]. Until now, techniques for trusted software protection are implemented after the design and implementation of the architecture is already completed. However, it is clear that some architectures are more vulnerable than others to specific types of attacks. Also, different programs are subject to different attacks that exploit different vulnerabilities. All these constraints can be simultaneously analyzed using techniques from hardware IP watermarking to ensure the creation of secure reusable hardware and software IP.

Semantic Hardware and Software IP Protection
All existing hardware IP protection techniques currently employ only syntax information for defining its objectives and accomplishing its tasks. At the same time, companies are spending hundreds of millions of dollars and sometimes even billions for their patent portfolios and their enforcement. There are also a number of very powerful reverse engineering techniques [20, 21]. There are even a number of techniques that are able to characterize chips using nondestructive techniques and by only employing remotely applied measurements. We believe that the use and processing of semantics can fundamentally alter and greatly improve the effectiveness of IP protection. These goals would also require conceptually new ideas and techniques. Most likely the first semantic IP protection techniques will have narrow objectives and target very specific pieces of semantic knowledge. Semantic hardware and software IP protection as well semantic protection in other domains are very difficult tasks. However, at the same time they are crucial for effective IP protection.

8 Conclusion

Hardware watermarking is a process of creating hardware IP in such a way that created artifacts contain undisputable proofs of authorship. The main flow of hardware IP watermarking consists of watermarking embedding, extraction, and steps for enforcement of digital rights. There are a number of augmenting and optional steps such as watermark enforcement and calculation of properties of the

watermarks such as the probability of coincidence, i.e. the likelihood of detection of non-intentional watermarks. We stress that the most critical aspects of hardware watermarking is to establish its global role as a hardware security technique and its relationship with other hardware security tasks. Finally, we proposed several very challenging and potentially practical and economically rewarding research and development directions including the creation of trusted hardware IP and the enforcement of semantic hardware IP rights.

References

1. Hong, I., Potkonjak, M.: Techniques for intellectual property protection of DSP designs. In: IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 3133–3136 (1998)
2. Lach, J., Mangione-Smith, W.H., Potkonjak, M: Fingerprinting digital circuits on programmable hardware. In: Information Hiding, pp. 16–31 (1998)
3. Lach, J., Mangione-Smith, W.H., Potkonjak, M.: FPGA fingerprinting techniques for protecting intellectual property. In: IEEE Custom Integrated Circuits Conference, pp. 299–302 (1998)
4. Qu, G., Potkonjak, M.: Analysis of watermarking techniques for graph coloring problem. In: IEEE/ACM International Conference on Computer-Aided Design, pp. 190–193 (1998)
5. Qu, G., Wong, J.L., Potkonjak, M.: Optimization-intensive watermarking techniques for decision problems. In: ACM/IEEE Design Automation Conference, New Orleans, LA, pp. 33–36 (1999)
6. Qu, G., Potkonjak, M.: Hiding signatures in graph coloring solutions. In: Information Hiding, pp. 348–367 (2000)
7. Qu, G., Wong, J.L., Potkonjak, M.: Fair watermarking techniques. In: Asia and South Pacific Design Automation Conference, Yokohama, Japan, pp. 55–60 (2000)
8. Qu, G., Potkonjak, M.: Fingerprinting intellectual property using constraint-addition. In: Design Automation Conference, pp. 587–592 (2000)
9. Qu, G., Potkonjak, M. Intellectual Property Protection in VLSI Design Theory and Practice. Kluwer. ISBN: 1-4020-7320-8 (2003)
10. Wong, J.L., Qu, G., Potkonjak, M.: Optimization-intensive watermarking techniques for decision problems. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **23**(1), 119–127 (2004)
11. Wolfe, G., Wong, J.L., Potkonjak, M.: Watermarking graph partitioning solutions. In: Design Automation Conference, pp. 486–489 (2001)
12. Kahng, A.B., Lach, J., Mangione-Smith, W.H., Mantik, S., Markov, I.L., Potkonjak, M., Tucker, P., Wang, H.J., Wolfe, G.: Constraint-based watermarking techniques for design IP protection. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **20**(10), 1236–1252 (2001)
13. Wolfe, G., Wong, J.L., Potkonjak, M.: Watermarking graph partitioning solutions. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **21**(10), 1196–1204 (2002)
14. Oliveira, A.L.: Robust techniques for watermarking sequential circuit designs. In: Design Automation Conference, pp. 837–842 (1999)
15. Oliveira, A.L.: Techniques for the creation of digital watermarks in sequential circuit designs. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **20**(9), 1101–1117 (2001)
16. Rashid, A., Asher, J., Mangione-Smith, W.H., Potkonjak, M.: Hierarchical watermarking for protection of DSP filter cores. In: IEEE Custom Integrated Circuits Conference, pp. 39–42 (1999)
17. Blythe, S., Fraboni, B., Lall, S., Ahmed, H., Deriu, U.: Layout reconstruction of complex silicon chips. IEEE J. Solid State Circuits **28**(2), 138–145 (1993)

18. Anderson, R., Kuhn, M.: Tamper resistance – a cautionary note. In: The Second Usenix Workshop on Electronic Commerce, pp. 1–11 (1996)
19. Anderson, R., Kuhn, M.: Low cost attacks on tamper resistant devices. In: Security Protocols, Springer, Heidelberg, pp. 125–136 (1998)
20. Torrance, R., James, D.: The state-of-the-art in IC reverse engineering. In: Cryptographic Hardware and Embedded Systems, pp. 363–381 (2009)
21. Torrance, R., James, D.: The state-of-the-art in semiconductor reverse engineering. In: ACM/EDAC/IEEE Design Automation Conference, pp. 333–338 (2011)
22. Van Le, T., Desmedt, Y.: Cryptanalysis of UCLA watermarking schemes for intellectual property protection. In: Information Hiding, pp. 213–225 (2003)
23. Kirovski, D., Potkonjak, M.: Efficient coloring of a large spectrum of graphs. In: Design Automation Conference, pp. 427–432 (1998)
24. VSI Alliance (1997) VSI Alliance Architecture Document: Version 1.0, Santa Clara, CA
25. Cox, I.J., Miller, M.L., Bloom, J.A.: In: Digital watermarking. Morgan Kaufmann Publishers. ISBN: 1-55860-714-5 (2001)
26. Abdel-Hamid, A.T., Tahar, S., Aboulhamid, E.M.: A survey on IP watermarking techniques. Des. Autom. Embed. Syst. **9**(3), 211–227 (2004)
27. Abdel-Hamid, A.T., Tahar, S., Aboulhamid, E.M.: IP watermarking techniques: survey and comparison. In: IEEE International Workshop on System-on-Chip for Real-Time Applications, pp. 60–65 (2003)
28. VSI Alliance (1997) Fall worldwide member meeting: a year of achievement. Santa Clara, CA
29. Kerckhoffs, A.: La Cryptographie Militaire. Journal des sciences militaires. **IX**, 5–38 (1883)
30. Wong, J.L., Majumdar, R., Potkonjak, M.: Fair watermarking using combinatorial isolation lemmas. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **23**(11), 1566–1574 (2004)
31. Fan, Y.C., Yang, H.Y., Tsao, H.W.: Direct access test scheme for IP core protection. In: IEEE Asia-Pacific Conference on Advanced System Integrated Circuits, pp. 262–265 (2004)
32. Fan, Y.C., Tsao, H.W.: Boundary scan test scheme for IP core identification via watermarking. IEICE Trans. Inf. Syst. **E88d**(7), 1397–1400 (2005)
33. Fan, Y.C.: Testing-based watermarking techniques for intellectual-property identification in SOC design. IEEE Trans. Instrum. Meas. **57**(3), 467–479 (2008)
34. Kean, T., McLaren, D., Marsh, C.: Verifying the authenticity of chip designs with the DesignTag system. In: IEEE International Workshop on Hardware-Oriented Security and Trust, pp. 59–64 (2008)
35. Ziener, D., Teich, J.: Power signature watermarking of IP cores for FPGAs. J. Signal Process Syst. Signal Image Video Technol. **51**(1), 123–136 (2008)
36. Becker, G.T., Kasper, M., Moradi, A., Paar, C.: Side-channel based watermarks for integrated circuits. In: IEEE International Workshop on Hardware-Oriented Security and Trust, pp. 13–14 (2010)
37. Yuan, L., Qu, G., Ghouti, L., Bouridane, A.: VLSI design IP protection: solutions, new challenges, and opportunities. In: First NASA/ESA Conference on Adaptive Hardware and Systems, pp. 469–476 (2006)
38. Kahng, A.B., Mantik, S., Markov, I.L., Potkonjak, M., Tucker, P., Huijuan, W., Wolfe, G.: Robust IP watermarking methodologies for physical design. In: Design Automation Conference, pp. 782–787 (1998)
39. Kahng, A.B., Kirovski, D., Mantik, S., Potkonjak, M., Wong, J.L.: Copy detection for intellectual property protection of VLSI designs. In: IEEE/ACM International Conference on Computer-Aided Design, pp. 600–604 (1999)
40. Chapman, R., Durrani, T.S.: IP protection of DSP algorithms for system on chip implementation. IEEE Trans. Signal Processing **48**(3), 854–861 (2000)
41. Megerian, S., Drinic, M., Potkonjak, M.: Watermarking integer linear programming solutions. In: Design Automation Conference (DAC), pp. 8–13 (2002)
42. Kirovski, D., Potkonjak, M.: Local watermarks: methodology and application to behavioral synthesis. IEEE Trans. CAD **22**(9), 1277–1284 (2003)

43. Koushanfar, F., Hong, I.K., Potkonjak, M.: Behavioral synthesis techniques for intellectual property protection. ACM Trans. Des. Automat. Electr. Syst. **10**(3), 523–545 (2005)
44. Charbon, E., Torunoglu, I.: Watermarking techniques for electronic circuit design. In: Lecture Notes on Computer Science 2613, pp. 147–169 (2000)
45. Kirovski, D., Hwang, Y.Y., Potkonjak, M., Cong, J.: Intellectual property protection by watermarking combinational logic synthesis solutions. In: IEEE/ACM International Conference on Computer-Aided Design, pp. 194–198 (1998)
46. Kirovski, D., Hwang, Y.Y., Potkonjak, M., Cong, J.: Protecting combinational logic synthesis solutions. IEEE Trans. CAD **25**(12), 2687–2696 (2006)
47. Khan, M.M., Tragoudas, S.: Rewiring for watermarking digital circuit netlists. IEEE Trans. CAD **24**(7), 1132–1137 (2005)
48. Cui, A.J., Chang, C.H., Tahar, S.: IP watermarking using incremental technology mapping at logic synthesis level. IEEE Trans. CAD **27**(9), 1565–1570 (2008)
49. Charbon, E., Torunoglu, I.: Watermarking layout topologies. In: ASP-DAC, pp. 213–216 (1999)
50. Narayan, N., Newbould, R.D., Carothers, J.D., Rodriguez, J.J., Holman, W.T.: IP protection for VLSI designs via watermarking of routes. In: IEEE International Asic/SoC Conference, pp. 406–410 (2001)
51. De Micheli, G.: Synthesis and Optimization of Digital Circuits. McGraw-Hill, New York (1994). ISBN 0070163332
52. Meguerdichian, S., Potkonjak, M.: Watermarking while preserving the critical path. In: Design Automation Conference (DAC), pp. 108–111 (2000)
53. Charbon, E.: Hierarchical watermarking in IC design. In: IEEE Custom Integrated Circuits Conference, pp. 295–298 (1998)
54. Charbon, E., Torunoglu, I.: Intellectual property protection via hierarchical watermarking. In: International Workshop on IP Based Synthesis and System Design (1998)
55. Cui, A.J., Chang, C.H., Zhang, L.: A hybrid watermarking scheme for sequential functions. In: IEEE International Symposium on Circuits and Systems, pp. 2333–2336 (2011)
56. Qu, G.: Publicly detectable watermarking for intellectual property authentication in VLSI design. IEEE Trans. CAD **21**(11), 1363–1368 (2002)
57. Torunoglu, I., Charbon, E.: Watermarking-based copyright protection of sequential functions. IEEE J. Solid State Circuits **35**(3), 434–440 (2000)
58. Cashwell, E.D., Everett, C.J.: A Practical Manual on the Monte Carlo Method for Random Walk Problems. Pergamon Press, New York (1959)
59. Abdel-Hamid, A.T., Tahar, S., Aboulhamid, E.M.: A public-key watermarking technique for IP designs. In: Design, Automation and Test in Europe, pp. 330–335 (2005)
60. Cui, A.J., Chang, C.H., Tahar, S., Abdel-Hamid, A.T.: A robust FSM watermarking scheme for IP protection of sequential circuit design. IEEE Trans. CAD **30**(5), 678–690 (2011)
61. Kirovski, D., Potkonjak, M.: Intellectual property protection using watermarking partial scan chains for sequential logic test generation. In: High Level Design, Test Verification (1998)
62. Cui, A.J., Chang, C.H.: Intellectual property authentication by watermarking scan chain in design-for-testability flow. In: IEEE International Symposium on Circuits and Systems, pp. 2645–2648 (2008)
63. Chang, C.H., Cui, A.J.: Synthesis-for-testability watermarking for field authentication of VLSI intellectual property. IEEE Trans. Circuits I **57**(7), 1618–1630 (2010)
64. Ziener, D., Baueregger, F., Teich, J.: Multiplexing methods for power watermarking. In: Hardware-Oriented Security and Trust (HOST), pp. 36–41 (2010)
65. Caldwell, A.E., Hyun-Jin, C., Kahng, A.B., Mantik, S., Potkonjak, M., Gang, Q., Wong, J.L.: Effective iterative techniques for fingerprinting design IP. In: Design Automation Conference, pp. 843–848 (1999)
66. Caldwell, A.E., Choi, H.J., Kahng, A.B., Mantik, S., Potkonjak, M., Qu, G., Wong, J.L.: Effective iterative techniques for fingerprinting design IP. IEEE Trans. CAD **23**(2), 208–215 (2004)

67. Lach, J., Mangione-Smith, W.H., Potkonjak, M.: Fingerprinting techniques for field-programmable gate array intellectual property protection. IEEE Trans. CAD **20**(10), 1253–1261 (2001)
68. Chang, C.H., Zhang, L.: A blind dynamic fingerprinting technique for sequential circuit intellectual property protection. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **33**(1), 76–89 (2014)
69. Agrawal, V., Cheng, K.-T.: Finite state machine synthesis with embedded test function. J. Electron. Test. **1**(3), 221–228 (1990)
70. Chaum, D.: Blind signatures for untraceable payments. In: Advances in Cryptography, pp. 199–203 (1982)
71. NIST Secure Hash Standard (SHS) (2012) http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf
72. Lofstrom, K.: System for providing an integrated circuit with a unique identification. US Patent No. 6,161,213. 12 (2000)
73. Koushanfar, F., Potkonjak, M.: CAD-based security, cryptography, and digital rights management. In: ACM/IEEE Design Automation Conference, pp. 268–269 (2007)
74. Alkabani, Y.M., Koushanfar, F.: Active hardware metering for intellectual property protection and security. In: The 16th Usenix Security Symposium, pp. 291–306 (2007)
75. Alkabani, Y., Koushanfar, F., Potkonjak, M.: Remote activation of ICs for piracy prevention and digital right management. In: IEEE/ACM International Conference on Computer-Aided Design, pp. 674–677 (2007)
76. Potkonjak, M., Nahapetian, A., Nelson, M., Massey, T.: Hardware Trojan horse detection using gate-level characterization. In: ACM/IEEE Design Automation Conference, pp. 688–693 (2009)
77. Sheng, W., Meguerdichian, S., Potkonjak, M.: Gate-level characterization: foundations and hardware security applications. In: Design Automation Conference, pp. 222–227 (2010)
78. Wei, S., Potkonjak, M.: Integrated circuit security techniques using variable supply voltage. In: Design Automation Conference, pp. 248–253 (2011)
79. Wei, S., Koushanfar, F., Potkonjak, M.: Integrated circuit digital rights management techniques using physical level characterization. In: ACM Workshop on Digital Rights Management, Chicago, IL, USA, pp. 3–14 (2011)
80. Wei, S., Nahapetian, A., Nelson, M., Koushanfar, F., Potkonjak, M.: Gate characterization using singular value decomposition: foundations and applications. IEEE Trans. Inf. Forensics Secur. **7**(2), 765–773 (2012)
81. Wei, S., Wendt, J.B., Nahapetian, A., Potkonjak, M.: Reverse engineering and prevention techniques for physical unclonable functions using side channels. In: Design Automation Conference, pp. 1–6 (2014)
82. Zheng, J.X., Li, D., Potkonjak, M.: A secure and unclonable embedded system using instruction-level PUF authentication. In: International Conference on Field Programmable Logic and Applications, pp. 1–4 (2014)
83. Zheng, J.X., Potkonjak, M.: A digital PUF-based IP protection architecture for network embedded systems. In: ACM/IEEE Symposium on Architectures for Networking and Communications Systems, Los Angeles, CA, USA, pp. 1–2 (2014)
84. Zheng, J.X., Potkonjak, M.: Securing netlist-level FPGA design through exploiting process variation and degradation. In: ACM/SIGDA International Symposium on Field Programmable Gate Arrays, Monterey, CA, USA, pp. 129–139 (2012)
85. Zheng, J.X., Chen, E., Potkonjak, M.: A benign hardware Trojan on FPGA-based embedded systems. In: International Conference on Field Programmable Logic and Applications, pp. 464–470 (2012)
86. Rajendran, J., Sam, M., Sinanoglu, O., Karri, R.: Security analysis of integrated circuit camouflaging. In: ACM SIGSAC Conference on Computer & Communications Security, Berlin, Germany, pp. 709–720 (2013)
87. Wendt, J.B., Koushanfar, F., Potkonjak, M.: Techniques for foundry identification. In: Design Automation Conference, pp. 1–6 (2014)

88. Mosteller, F.: Fifty Challenging Problems in Probability with Solutions. Courier Dover Publications (1987)
89. Wei, S., Nahapetian, A., Potkonjak, M.: Robust passive hardware metering. In: IEEE/ACM International Conference on Computer-Aided Design, pp. 802–809 (2011)
90. Sheng, W., Nahapetian, A., Potkonjak, M.: Quantitative intellectual property protection using physical-level characterization. IEEE Trans. Inf. Forensics Secur. **8**(11), 1722–1730 (2013)
91. Dabiri, F., Potkonjak, M.: Hardware aging-based software metering. In: Design, Automation & Test in Europe Conference & Exhibition, pp. 460–465 (2009)
92. Potkonjak, M.: Usage metering based upon hardware aging. US Patent 8,260,708 (2012)
93. Potkonjak, M.: Synthesis of trustable ICs using untrusted CAD tools. In: Design Automation Conference, pp. 633–634 (2010)
94. Koushanfar, F., Fazzari, S., McCants, C., Bryson, W., Sale, M., Song, P.L., Potkonjak, M.: Can EDA combat the rise of electronic counterfeiting? In: Design Automation Conference, pp. 133–138 (2012)
95. Collberg, C., Thomborson, C.: Software watermarking: models and dynamic embeddings. In: ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Antonio, TX, USA, pp. 311–324 (1999)
96. Palsberg, J., Krishnaswamy, S., Kwon, M., Ma, D., Shao, Q., Zhang, Y.: Experience with software watermarking. In: Annual Computer Security Applications Conference, pp. 308–316 (2000)
97. Kirovski, D., Drini, M., Potkonjak, M.: Enabling trusted software integrity. In: Architectural Support for Programming Languages and Operating Systems, pp. 108–120 (2002)