Provably Secure Obfuscation of Diverse Watermarks for Sequential Circuits

Farinaz Koushanfar
Electrical and Computer Engineering
William Marsh Rice University
Houston, TX
Email: farinaz@rice.edu

Yousra Alkabani
Computer Science
William Marsh Rice University
Houston, TX
Email: yousra@rice.edu

***Abstract*—This paper presents a provably secure method for embedding multiple watermarks in sequential designs. A number of different watermarks signed with the IP owner's secret key from a public key cryptography system are generated. The owner's watermarks are then dissembled into the states and transitions of the original sequential design. Hiding the multiple watermarks in the states and transitions is shown to be an instance of obfuscating a multi-point function with a generalized output. We draw on the theoretical cryptographic results of provable obfuscation of this function family to build a secure sequential multi-watermark system by construction. An iterative synthesis method for integrating the collection of watermarks to the original design is introduced. Analysis of watermark properties and the attack resiliency of the new multiple watermarking construction is presented. Experimental evaluations on benchmark circuits demonstrate practicality and low overhead of the new provably secure multiple watermarks construction method.**

I. Introduction

Watermarking of hardware intellectual property (IP) is the process of embedding information into a digital design in a way that is difficult to tamper, forge, or remove. The embedded information is used for secure identification of the design's authenticity and as a proof of ownership. Secure watermarks can be used in court cases as an IP ownership proof. Furthermore, the addition of watermarks would deter unauthorized usage of the IP, thus automatically decreasing the rate of IP piracy.

Design houses commonly source third-party IP cores to manage the increasing design complexity and time-to-market pressure. IP design and reuse is particularly important for small and medium sized design houses and system-on-a-chip designers. Typically a design house is specialized in a subset of IPs, e.g., communications, security, buses, or memory, and the company's revenue is based on the IP value. A major issue in this business model is illegal usage, theft, or piracy of the design IPs. As an example, imagine the following scenario: a design house A collects the royalty to allow company B the rights of using its intellectual property (denoted by IP_A) for the design D_{1B}. Company B, with access to the core IP_A, would also reuse the core in its design D_{2B} without paying the proper royalties. Such violations are hard to track in the final products because of the clandestine nature of the integrated circuits. In a long run, illegal IP usage not only would result in extinction of smaller design companies, but also it would taint a healthy and productive design and reuse business model.

Since design watermarking is an effective and efficient method for protecting the rights of IP owners, a number of interesting and efficient watermarking methods were proposed in the literature [1], [2], [3], [4], [5], [6], [7], [8]. Watermarking can be done at the different steps during synthesis and layout processes. It has been demonstrated that embedding several watermarks instead of one, not only makes the method more secure against removal attacks, but also creates additional means for ownership proof, once a subset of watermarks is revealed [5]. One particularly interesting class of watermarks are those that embed the designer (author) signature in the sequential structure of the design [2], [3], [4]. In this way, not only the hard IP, but also a firm IP (netlist or HDL code) can be protected.

The sequential watermarks are usually inserted by manipulating the states and transitions of the finite state machine, representing the functional specification of the design. Even though the previous research have suggested a number of methods for hiding the watermarks inside the state and transitions of finite state machine, the earlier published results in this domain only suggested heuristic approaches to the problem without a strong proof of security.

This paper introduces the first secure method with cryptographic proof for embedding multiple watermarks in the sequential designs. The contributions are:

- We Create the first provably secure method for sequential circuit watermark embedding. The new distributed watermarking method inserts multiple tamper-resilient disjoint watermarks in the design.
- We introduce a watermark construction and embedding method where watermarking corresponds to concealing a multi-point function with a generalized output. Our method is secure as the family of point functions has been demonstrated to be efficiently and provably obfuscatable.
- A number of attacks and security of the method against them are discussed. We also outline a number of measures that can be taken for further design protection.
- An algorithm is devised for automatic integration of the watermarks into the sequential design during synthesis.
- Experimental results for watermarking sequential bench-

978-1-4244-7812-5/10/$26.00 ©2010 IEEE

marks demonstrate negligible power, area, and timing overheads and efficiency of the new methods.

The related work and background are discussed in Sections II-III. The provably secure watermark embedding algorithm is devised in Section IV. Attack resiliency is discussed in Section V. We show the experimental results in Section VI and conclude in Section VII.

II. Related Work

Watermarking is an efficient mean for protecting the ownership of the design IP. Watermarking at different levels of abstraction is useful and necessary, in particular for cases where more than one party is considered and the IP infringement tracking is more difficult. Therefore, a number of methods for watermarking digital designs were proposed at different levels of design abstraction. For example, during the physical design, the watermark can be embedded as a set of constraints on the topology, position, orientation, and relative placement. As another example, watermark can be inserted as constraints within the synthesis optimizations including register assignment and scheduling. The authors in [1] provide an excellent summary of watermarking at different levels of abstraction.

We focus on sequential circuit watermarking at the behavioral level. This watermarking mechanism inserts additional states and transitions in the finite state machine of the design [4], [2], [3]. While the earlier work has demonstrated watermarking mechanisms by state and transition addition, no cryptographic guarantees were provided. We show a provably secure watermarking method, by posing the watermarking problem as an instance of point function obfuscation. While constructing our watermark, we ensure that the efficient obfuscation properties are satisfied. Another new aspect of the current paper is addition of multiple disjoint watermarks that has not been done for sequential designs earlier but provide a stronger protection [9]. Note that family of sequential circuit watermarks can also be used for protecting FPGA IPs [10].

An important problem in computer security is if a program can be obfuscated, i.e., if the code can be made unintelligible while preserving its functionality. A number of adhoc techniques for program obfuscation were proposed in the past several decades. About a decade ago, the first theoretical study of obfuscation was initiated by Barak et al. [11]. Not only did they attempt at formally defining the problem, but also they demonstrated a "generic" obfuscator for all programs does not exist. In 2004, Lynn et al. demonstrated the first positive theoretical results for obfuscation of specific function families [12]. Since then, a few other researchers have pursued obfuscation of specific function families and their generalizations [13], [14]. In earlier work, we demonstrated the first set of applications of obfuscation by hiding states in the finite state machine for hardware IP protection, activation, and disabling [15], [16], [17]. However, to the best of our knowledge this paper is the first to use the theoretical obfuscation results for hardware IP protection.

III. Background

To be self-contained, we provide the necessary background of terms and definitions used in the remainder of the paper.

Finite state machine (FSM). FSM is a discrete dynamical model that maps input vector sequences into output vector sequences. FSM model is widely used in automata theory, formal verification, high level design, and in cryptography. FSM can be used to model any regular sequential function and is expressed in different formats, e.g., Verilog HDL or VHDL. We use the state transition graph (STG) to represent the states and transitions and output functions of the FSM. On the STG, the nodes correspond to the states and the edges define the input/output conditions for a state-to-state transition.

Hardware IP format. Typically, a hardware IP is classified as hard, firm, or soft based on the degree of freedom the user has in modifying it. Hard IP, is commonly in form of a routed layout and cannot be changed by the user. In contrast, soft IP can be transformed in many ways. For example, a soft IP can be synthesized, floorplanned, mapped, and laid out. A firm IP is a middle level between the two formats. For example, a netlist that is not mapped to the specific technology is a firm IP. The focus of this paper is on firm IP protection. The assumption is that the STG is kept by the IP rights owner and is not transfered to other entities who have the firm IP [15].

IV. Provably secure watermarking

Sequential watermarking methods commonly add the IP owner's signature by creating additional transitions (edges) to the design's STG. The new edges may be either incident to the original states, or they may be incident to some added states. The watermark is inserted by assigning transition inputs to a sequence of one or more added transitions that are not traversed during the normal design operations. The inputs corresponding to added transitions are only known to the watermark designer (or the IP rights owner). If the IP owner embeds the watermark such that it corresponds to her signature, then only the designer can supply the inputs that would take the sequential circuit through the watermark transitions. Therefore, she could provide a proof of ownership.

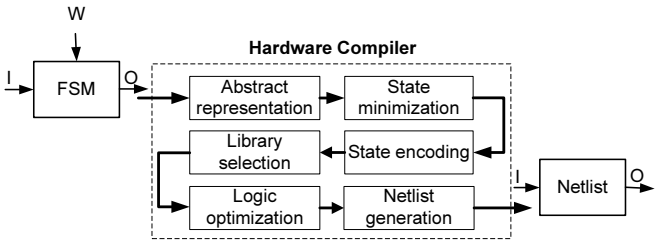

Fig. 1. Block diagram of the synthesis flow (hardware compilation).

Let us take a look back into the hardware design and watermark embedding flow steps demonstrated in Figure 1. In essence, as shown on the figure, this design flow is a hardware compiler. The input program (circuit) to the compiler is the high level sequential design specification (modeled

by the FSM) plus the owner's watermark (denoted by W). The output program (circuit) is a compilation of high level design specifications to low level logic components (netlist). Both circuits have the same input/output mapping, but their internal structures are different. The goal of watermarking is to hide information in high level circuit, such that the hidden information cannot be retrieved or removed during the circuit's regular functional behavior. The covered watermark can only be retrieved by entities who have the exact knowledge of the hidden secret placement in the high level functional specification. There should be a negligible probability that others would be able to find or forge this hidden watermark.

In effect, we want the above hardware compiler to act as an obfuscator over the hidden watermark. An obfuscation is informally defined as an efficient probabilistic compiler O that transforms a source program (circuit) F into a new program $O(F)$ (also a circuit) such that:

1) $O(F)$ has approximately the same observable input/output behavior as F;
2) For an input vector, the speed of computing the corresponding output by $O(F)$ compared with F is at most polynomially different (slower or faster);
3) $O(F)$ is substantially "less intelligible" than F;

While we will not delve into formal definitions in this paper, we emphasize the following two key metrics that are essential for a formal definition. In the first condition, the term approximate functionality has to be clarified. In formal definitions given in [11], this approximate functionality is determined such that the input/output relationships of F and $O(F)$ are the same with an extremely high probability. The term less intelligible (less readable) in the last condition, has been subject of much debate. Some researchers have opted to use the "virtual blackbox model", where no information about the internals of F can be obtained by studying $O(F)$ other than the input/output behavior of the system [11], [12]. Others have introduced more realistic definitions, where some information could be leaked from the obfuscator [14].

Theoretical research in cryptography has demonstrated that although a generic program obfuscator does not exist, there are specific classes of functions that can be efficiently obfuscated (i.e., satisfying above definitions). One such family of functions are "point functions" [12]. A *point function* $f_{\alpha,\beta} : \{0,1\}^k \rightarrow \{0,1\}^{s(k)}$, with a *generalized output* can be defined as:

$$f_{\alpha,\beta}(x) = \begin{cases} \beta & \text{if } x = \alpha \\ \perp & \text{otherwise.} \end{cases} \quad (1)$$

In a point function, there is a very large state-space of possibilities. However, only one point in this very large state-space is able to generate the required output string. The output is called generalized since its value is more than one bit. A prime example of a point function that is widely used in security applications is a password. A password is considered to be secure, since the probability of guessing the right one is extremely low. This is indeed an obfuscation, where the users have access to the program and can supply their input and check the output, but have an extremely low chance of figuring out the hidden information.

Theorem. For random oracles R : $\{0, 1\}^* \rightarrow \{0, 1\}^{2k}$, if we define a point function ($f_{\alpha,\beta}$) where the value is 1 only for one point, then O is an obfuscator according to the definition outlined earlier in this section [12].

Lemma. Adding a watermark by a set of sequential edge transitions to the STG such that a set of specific vertices would be visited by traversing on the edges, forms a point function with a generalized output and is thus obfuscatable.

Proof. Assume that the modified STG with the watermark edge sequence is not a point function. Now, there should be another set of transitions that would traverse the same states in response to an input. However, such edges do not exist by our graph construction. Therefore, the embedded watermark forms a point function on the graph and is obfuscatable.

In other words, if one designs the state-space of possibilities for watermark transitions to be extremely low, then the watermark cannot be guessed. Our watermarking method of choice is similar to the earlier sequential watermark embedding mechanisms in that the watermark transitions the design through a series of hidden states until they reach a final hidden state. The major novelty is the obfuscation guarantee by point function properties outlined in this section, and secure construction methods that would be presented in the next two subsections.

A sensitive issue is information leakage by the FF content. That is, how much information about the watermark can be collected by having access to the scan chains reading the FF values? Would this information help a potential adversary in guessing the transitions required for reaching the watermarked states? In our construction method, we ensure that the transitions inputs are independent of the FF values, and the watermark sequences are not shared with the other states, so this leakage would not be a problem.

Another key issue that we address in this paper is multiple watermarking. If only one watermark is inserted, the first time a proof of ownership is provided, the watermark will be publicly known and the method would be more vulnerable to forging attacks. To overcome this limitation, and to distribute the watermark across the design, we embed multiple watermarks in the design [9], [5]. There are two questions: (i) Would the results for obfuscating a point function carry on to this case? (ii) Would revealing one transition path signature leak information about the other signatures in the design?

To address multiple watermarking, let us discuss a *multi-point function with a generalized output* defined as follows:

$$f_{(\alpha_1,\beta_1),\ldots,(\alpha_t,\beta_t)}(x) = \begin{cases} \beta_i & \text{if } x = \alpha_i \\ \perp & \text{otherwise.} \end{cases} \quad (2)$$

Lemma. Adding multiple transitions to the hidden states on STG forms a multi-point function with a generalized output.

The proof can be derived similarly to the single point function case. Fortunately, the multi-point function family has also been demonstrated to be obfuscatable under the random oracle model [12]. The analogy here is a system where instead

of one password, there are multiple passwords that can be use for traversing the different transitions and states in state-spac

A. Watermark and signature creation

The watermark is a special state that is only reached b the original design when a designer constructed signature applied to the circuit as a sequence of primary inputs. In th subsection we discuss our watermark construction metho The steps for creating the signatures are as follows.

1) The design owner selects m signature messages; m the number of watermarks.
2) A public/private key pair of a public key crypto-syste is selected by the designer.
3) The m messages are signed by designer's private key.
4) A standard hash function is used for obtaining a fixe length digest message.
5) New edges are created on the STG and the message digest is assigned as traversing input to new edges.

Note that the state encoding is an independent stage that is done after adding the transitions. This implies that there is no dependence between the new edges and the contents of the FFs. Thus, knowing an encoding does not reveal any information about the edges.

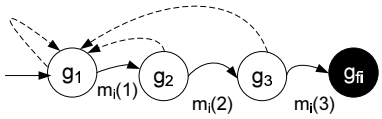

Fig. 2. The watermark STG with signature m_1 integrated as primary input sequence.

Assume we have n sets of input transitions (signatures) $m_i = (m_i(1), m_i(2), m_i(3))$, where $1 \leq i \leq n$ that are divided into three segments each. We generate a watermark STG as shown in Figure 2 as follows.

- For each signature m_i, create a unique path that goes from a starting node to a final node. In our example, this path is g_1, g_2, g_3, g_{fi} for m_i. These paths are marked with solid lines on Figure 2.
- At any intermediate state (in our case g_2 and g_3), if the expected part of the signature is not input, a transition is taken back to g_1. These transitions are shown as dashed lines in Figure 2.
- Once a final state is reached (g_{fi}, the STG stays in this final state until reset. A final state is only reached when an input corresponding to a signature is applied to the circuit. The sequence of states traversed and the final state reached using a signature represent the watermark.

B. Watermark embedding

Once the watermark STG is constructed, it will be merged with the original STG. The watermark STG can be implemented in don't care states of the original FSM and encoding of these states can be used to check the watermark [3]. However, in this work we opt to use iterative synthesis to merge the two STGs. In this way there is no need to extract the STG from designs available in other formats.

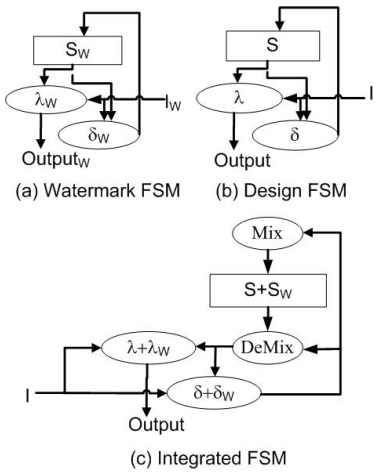

Fig. 3. The integration of (a) the watermark FSM, (b) the design FSM, into (c) the integrated FSM.

In our implementation, the watermark STG and the original one initially synthesized separately. Then, the resulting FSMs are merged together. Figure 3(a) and (b) show the watermark FSM and the original FSM after synthesis respectively. The FFs representing the state elements of the watermark FSM and the original FSM are shown as S_W and S, respectively. The combinational parts to compute the next state and the output for the watermark FSM are represented by δ_W and λ_W respectively, while their counterparts in the original design are shown as δ and λ.

Figure 3(c) shows how the two FSMs are combined to generate the integrated FSM of the watermarked design. The integrated FSM is constructed by merging the state elements $S+S_W$ and merging the combinational logic parts $\delta+\delta_W$ and $\lambda+\lambda_W$. However, two combinational logic circuits are added shown as Mix and $DeMix$. The Mix circuit takes the output from δ_W representing the next state of the watermark FSM and permutes the order of the state elements such that the state elements used by the watermark FSM and the original FSM are different. Each state in the watermark FSM corresponds to one permutation except g_1 that is associated by multiple permutations that are a function of the next state of the original design. The $DeMix$ circuit reorders the output from the state elements to maintain the correct operation of the combinational part. The whole integrated FSM is resynthesized and optimized to interweave the logic.

C. Watermark verification

The designer can use her knowledge of the watermark FSM and the original design to compute the expected set of states to be traversed when applying each signature. In fact, the designer does not need to compute the whole STG. It is sufficient to check the subset of flipflops corresponding to the watermark FSM. This subset will change every time a part of the signature is applied because a new state is reached. The watermark FSM is much smaller than the design FSM

and is easy to compute if the STG is known without being obfuscated in the design. Note that even if one watermark is revealed, it would not uncover any information about the other watermarks. This is because the paths corresponding to different signature do not intersect by our STG construction.

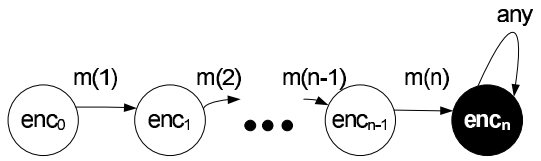

Fig. 4. Verification sequence for signature $m = (m(1), m(2), ..., m(n))$.

Figure 4 shows the verification sequence for a signature $m = (m(1), m(2), ..., m(n))$ that is divided into n segments. To verify the ownership, the prover has to predict the values of enc_0 to enc_n and show that after applying the signature enc_n is reached.

V. Watermark properties and attack resiliency

A watermark must satisfy a set of following properties to ensure security and resiliency against attacks:

- *Unobtrusiveness:* The watermark should be invisible to the functional circuit. Its presences should not interfere with the regular operation of the design.
- *Robustness:* The embedded watermark should be extremely difficult (almost impossible) to remove. It should not be distinguishable and removable from the knowledge available to entities who do not have the knowledge of where in the design it is embedded. In particular, the watermark should be robust to: (1) the common transformations and optimizations performed on an IP and during the design flow after the watermark embedding step; (2) collision of watermarks, such that multiple individuals who create a signature for their designs each have a unique set of characteristics and can be specifically identified; (3) forgery of the signature, such that nobody else but the original designer can use the watermark to prove its ownership.
- *Unambiguity:* Retrieving the watermark must be conclusive in proving the ownership.
- *Universality:* The same watermarking method must be applicable to all common sequential circuits.

In our attack model, every time performing an adversarial task is as hard as redesigning the IP the attack is considered to be invalid. After all, if the adversary has the resources to build the design, they would not need to illegally use the IP. We demonstrate that the watermark properties listed above are satisfied by our watermarking construction algorithm:

- *Unobtrusiveness:* The watermark would not cause an interference since the transitions taken by the watermark on the STG are clearly separated from the ones taken during the normal IP's operational states and transitions.
- *Robustness:* The three listed robustness conditions are present in our method: (1) watermark is present in the netlist, so all optimizations and transformations after this stage would contain the watermark; (2) the degree of freedom and the length of edge weights and the randomness of weight values because of hashing and cryptographic signing ensure that the watermarks of different designers would not collide; (3) forgery of the signatures is also nulled since the watermark sequence is long and random and therefore, brute-force attacks for learning all the states and transitions of the STG (including the watermark that is a multi-point function) is computationally infeasible. The probability of a forger finding states and transitions according to a new signature is extremely low and can be further lowered by increasing the size of the STG inputs and its state-space.
- *Unambiguity:* The probability of finding the multi-point function is low by our watermark construction method. Therefore, the probability of finding a valid watermarking sequence which also corresponds to the cryptographic signature of the owner is almost negligible. The retrieval of a watermark that corresponds to the designer's signature is a strong proof of ownership.
- *Universality:* The FSM based watermarking can be used for all common sequential designs.

Note that the information leakage by FF scanning would not help an attacker as was mentioned in Section IV according to the best possible obfuscation model [14]. The information hiding in the state transitions would still be secure, even though the state encoding can be scanned. In multiple watermarking, the different watermarks do not share states or transitions. Therefore, a revealed watermark would not uncover information about other hidden signatures.

VI. Experimental results

In this section we study the overhead of our watermarking method. We evaluate the overhead on standard MCNC'91 benchmarks. For the MCNC'91 benchmarks we use Matlab to generate the watermark FSM in KISS format and use ABC for the iterative synthesis process.

TABLE I
The synthesized benchmarks without any watemarking.

circuit	PI	PO	reg	area	delay	power
s641	35	23	19	539	97.6	1560.6
s713	35	23	19	591	100	1670.7
s1423	17	5	74	1164	92.4	4882.7
s5378	35	49	164	4212	32.2	12459.4
s9234	36	39	211	7971	75.8	19385.5
s15850	14	87	597	13659	116	40002.7
s13207	31	121	669	11241	85.6	37843.6
s38584	12	278	1452	32910	94.2	112706.8
s35932	35	320	1728	28269	299.4	122048.4

Table I shows the circuits used for evaluation. The first column shows the circuit name ($cicuit$). Number of primary inputs and outputs (PI and PO) are shown in the second and third columns. The fourth column presents the number of FFs (reg). The original $area$ in the number of literals, delay of the critical paths in *ns* and power in *mW* are shown in the

TABLE II
OVERHEAD OF ADDING 4, 8, AND 16 WATERMARK SIGNATURES.

	4 watermarks			8 watermarks			16 watermarks		
circuit	**%area**	**%delay**	**%power**	**%area**	**%delay**	**%power**	**%area**	**%delay**	**%power**
s641	236.2	4.1	265.7	568.8	2	650.4	1,352.50	0.4	1,530.50
s713	216.6	4	250.6	517.9	2	607.5	1,233.50	0.4	1,429.60
s1423	178	5.4	142.9	381	5.8	302.4	859	4.3	658.1
s5378	30.4	12.4	33.6	72.7	6.2	81.5	173.1	1.2	191.7
s9234	18.3	0.5	26.5	40.2	3.4	53.6	89.5	4	124.4
s15850	14.6	2.4	16.2	37.2	2.2	40.4	86.8	2.6	91
s13207	14.4	0	13.8	39.1	0	36.9	71.5	0	64.1
s38584	7.2	0	6.8	16.9	0	15.7	39.5	0	35.1
s35932	4.5	1.3	3.4	10.8	0.7	8.3	25.8	0.1	19.6

last three columns. Table II shows the percentage overhead in area, delay of critical path, and power introduced by adding four watermarks (columns 2-4), eight watermarks (columns 5-7), and sixteen watermarks (columns 8-10).

We guarantee that each design watermark is cryptographically secure by ensuring that the sum of the number of primary inputs and registers is at least 64. This number can be easily extended to larger sizes (e.g., 128). The level of security achieved by this number is similar to a cryptographic system with a key size of the same number of bits.

To achieve this security guarantee, for the first two benchmarks one needs to add at least 10 extra FFs. On smaller designs, our method incurs a higher overhead compared with the earlier sequential watermarks [4], [3], [2]. It can be seen that the overhead of our watermarking decreases as the size of the original design increases. On average the area and power overheads of the benchmarks are about 80%, 190%, and 450% in case of 4, 8, and 16 watermarks respectively. The impact on the critical path does not exceed an average of 3.5% in all cases. If we exclude the very small benchmarks, the area and power overheads would drop to an average of 15%, 37%, and 85% for the different number of watermarks. Also, the delay does not exceed an average of 2.8%. Note that the finite state machine (control part of the circuitry) typically represents less than 1% of the overall design, so even doubling this part does not significantly add to the design overhead [15].

VII. CONCLUSION

This paper presented a new provably secure sequential watermarking method. A set of cryptographically strong signatures was generated by the IP owner and embedded within the design's states and transitions. We showed that hiding the signatures in sequential design structure is an instance of a multi-point function with a generalized output. The theoretical results of obfuscating the family of point functions were used for making a secure watermarking method by construction. The hardware synthesis and optimization steps which generates the circuit netlist from high level specification implement the obfuscating transformations. We discussed automating multiple watermarks integration by iterative synthesis. Security and attack resiliency of the new method were discussed. Practicality and efficiency of the method were demonstrated by evaluations on sequential benchmark circuits.

REFERENCES

[1] G. Qu and M. Potkonjak, *Intellectual Property Protection in* VLSI *Design*. Kluwer Academic Publisher, 2003.

[2] A. Oliveira, "Techniques for the creation of digital watermarks in sequential circuit designs," *IEEE Trans. CAD of Integrated Circuits and Systems*, vol. 20, no. 9, pp. 1101–1117, 2001.

[3] L. Yuan and G. Qu, "Information hiding in finite state machine," in *Information Hiding Workshop*, 2004, pp. 340–354.

[4] I. Torunoglu and E. Charbon, "Watermarking-based copyright protection of sequential functions," *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 35, no. 3, pp. 434–440, 2000.

[5] D. Kirovski and M. Potkonjak, "Local watermarks: Methodology and application to behavioral synthesis," *IEEE Trans. CAD of Integrated Circuits and Systems*, vol. 22, no. 9, pp. 1277–1284, 2003.

[6] F. Koushanfar, I. Hong, and M. Potkonjak, "Behavioral synthesis techniques for intellectual property protection," *ACM Trans. Design Automation of Electronic Systems (TODAES)*, vol. 10, no. 3, pp. 523–545, 2005.

[7] G. Qu and M. Potkonjak, "Analysis of watermarking techniques for graph coloring problem," in *International Conference on Computer-Aided Design (ICCAD)*, 1998, pp. 190–193.

[8] D. Kirovski, Y. Hwang, M. Potkonjak, and J. Cong, "Intellectual property protection by watermarking combinational logic synthesis solutions," in *International Conference on Computer-Aided Design (ICCAD)*, 1998, pp. 194–198.

[9] I. Brown, C. Perkins, and J. Crowcroft, "Watercasting: Distributed watermarking of multicast media," in *Workshop on Networked Group Communication (NGC)*, 1999, pp. 286–300.

[10] J. Lach, W. Mangione-Smith, and M. Potkonjak, "Signature hiding techniques for fpga intellectual property protection," in *International Conference on Computer-Aided Design (ICCAD)*, 1998, pp. 186–189.

[11] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang, "On the (im)possibility of obfuscating programs," in *Annual International Cryptology Conference on Advances in Cryptology (CRYPTO)*, 2001, pp. 1–18.

[12] B. Lynn, M. Prabhakaran, and A. Sahai, "Positive results and techniques for obfuscation," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 2004, pp. 20–39.

[13] S. Hohenberger, G. Rothblum, A. Shelat, and V. Vaikuntanathan, "Securely obfuscating re-encryption," in *Theory of Cryptography Conference (TCC)*, 2007, pp. 233–252.

[14] S. Goldwasser and G. Rothblum, "On best-possible obfuscation," in *Theory of Cryptography Conference (TCC)*, 2007, pp. 194–213.

[15] Y. Alkabani and F. Koushanfar, "Active hardware metering for intellectual property protection and security," in *USENIX Security Symposium*, 2007, pp. 1–16.

[16] Y. Alkabani, F. Koushanfar, and M. Potkonjak, "Remote activation of ICs for piracy prevention and digital right management," in *International Conference on Computer-Aided Design*, 2007, pp. 674–677.

[17] Y. Alkabani and F. Koushanfar, "N-variant IC design: methodology and applications," in *Design Automation Conference (DAC)*, 2008, pp. 546–551.