Watermarking-Based Copyright Protection of Sequential Functions

Ilhami Torunoglu and Edoardo Charbon, *Member, IEEE*

***Abstract*—Watermarking is one of several techniques available today to deter copyright infringement in electronic systems. The technique consists of implanting indelible stamps in the circuit's inner structure, while not disrupting its functionality nor degrading its performance significantly. In this paper, a novel method is proposed for the creation of watermarks in regular sequential functions. This is an important class of functions, as it is the basis of most digital controllers. Algorithms are proposed for implanting robust watermarks to minimize the overhead and, ultimately, to reduce the impact on performance. Detection methods have been discussed in the presence of infringement attacks. The resilience of the method in several tampering regimes has been estimated. Examples illustrate the suitability of the approach.**

***Index Terms*—Copyright protection, digital design, finite state machines, watermarking.**

I. Introduction

TODAY, electronic systems are built in large part using stand-alone, individually packaged chips, assembled on ad hoc printed circuit boards. The industry is currently shifting to a new design paradigm based on the system-on-chip concept. Future systems will be assembled integrating several building blocks, so-called virtual components, on the same silicon substrate. Virtual components, associated to intellectual properties (IP's), will be designed by independent firms, possibly for a number of technologies and applications. To ensure that proper mechanisms exist to govern the exchange and management of IP's, a set of standards and interfaces are currently being defined [1].

One of the fundamental requirements to promote a practical system-on-chip design paradigm is that copyrights of the design and of its building blocks be safeguarded. In particular, it will become essential that the industry find ways to fight potential IP copyright infringement. Currently, design copyright laws are enforced by means of nondisclosure agreements and patents. However, the costs involved in preventing or containing IP infringement and tracking espionage, if at all possible, may be too high.

A promising alternative is deterrence. A possible such scheme requires the capability of effectively detecting and subsequently tracking IP infringement cases. This task can be accomplished by embedding a unique code, or *watermark*, exploiting the IP's unique features. Fundamental requirements for a watermark are that it be 1) transparent, i.e., not interfering with the design functionality, 2) robust, i.e., hard to remove or forge, and 3) detectable, i.e., easy to extract from the design. The process used for managing watermarks must not necessarily be proprietary, while the code used in the encryption process should be secret for any released IP.

Recently, watermarking has been applied to digital audio-visual IP's [2], [3]. The proposed techniques, though with small variations, essentially consist of superimposing a pseudorandom noise to the signal of the record. Such noise, though completely inaudible, can be easily detected via digital signal-processing methods.

Schemes based on watermarking have been recently proposed for electronic IP's as well. In [4] and [5], the watermark assumes the form of a extraneous circuit, hidden inside large field-programmable gate arrays (FPGA's). Such circuits are implemented in the FPGA's lookup tables and connected to the rest of the circuit without disrupting the original functionality.

In [6] and [7], we have proposed to incorporate several watermarks, distributed over all the abstraction levels of a given design. The techniques differ depending on the abstraction level to which they are applicable. At the physical design level, the watermark assumes the form of a set of topological constraints governing the relative position, orientation, and, possibly, scaling of the devices or gates of the circuit. At netlist and register-transfer level, constraints on the structure of a selected set of nets are used to represent the watermark.

Several authors have proposed to use other design constraints to implant watermarks. In [8], fixed placement and delay constraints implemented the watermark. In [9], a sequence of nodes in a multilevel logic function was permuted according to a seeded pseudorandom selection scheme.

In [10] and [11], schemes have been proposed to implant watermarks in regular sequential functions by modifying the original function in a structured fashion. In this paper, we will focus on this kind of watermarking scheme, due to the omnipresence of regular sequential functions in industrial designs and the resilience of the scheme against tampering at lower abstraction levels.

In the case that more than one party is involved in the creation of an IP, none of the above techniques alone guarantees that the infringements can be tracked. Watermarking should be performed simultaneously at various levels of abstraction [6]. The goal is to improve the robustness of the approach and to allow quick and accurate tracking of the last licensee, who ultimately caused the infringement.

At least two types of watermarking schemes exist. The first scheme, known as *active watermarking*, consists of integrating

Manuscript received July 19, 1999; revised October 19, 1999.

The authors are with Cadence Design Systems, Inc., San Jose, CA 95134 USA.

Publisher Item Identifier S 0018-9200(00)00537-0.

the watermark as a part of the design process, thus allowing the creation of an arbitrarily high number of uniquely watermarked designs. In the second scheme, known as *passive watermarking* or *fingerprinting*, one creates a unique and compact representation of a design at any abstraction level. This representation, known as *digital signature*, can be used to track infringement after it occurred by simply extracting the signature from an existing design and comparing it with the original one. To avoid false claims, a third-party organization should maintain a data base of all registered signatures for which protection is sought [12]. Both approaches are robust, since the deletion of the watermark results, with high probability, in the removal of wanted functionality.

IP protection based on watermarking consists of two phases: *synthesis* and *detection*. The synthesis phase is fully characterized by:

1) a set of algorithms translating design features onto a unique watermark;
2) P_u, the odds that an unintended watermark is detected in a design.

The detection phase is fully characterized by:

1) P_m, the probability of a miss;
2) $P_f = P_u$, the probability of a false alarm.

In this paper, we propose a set of algorithms for synthesizing watermarks in an important class of circuits, which implement *regular sequential functions*. In its most abstract form, the methodology can be described as follows. A regular sequential function is modified to generate a predictable output sequence when an unspecified input sequence is applied. In this context, the watermark is a pair of input/output sequences of symbols, which cannot occur during normal operation. Such sequences are hidden among "legal" input/output sequences, thus making it extremely time-consuming to track and remove them, with the risk of accidentally modifying intended functionality.

It is possible that the output sequence is defined for every possible input sequence even if the input sequence is an illegal one, as in the error handling case. In this case, by augmenting the input and/or output set, as one would do when adding some testing signals, some input/output sequences will not be defined. Using these undefined input/output sequences, one can insert the watermark into the regular sequential function.

The proposed methods fundamentally differ from recently proposed finite-state machine (FSM) watermarking techniques, which rely on topological watermarking. Topological watermarking consists of injecting an extraneous state/transition topology into the FSM without changing its behavior. Although it creates a unique watermark, the detection becomes a very hard problem due to the fact that the watermark detection problem is equivalent to an automated test pattern generation (ATPG) problem, which is known to be NP-complete [11].

In our approach, the detection problem becomes a very easy task. The existence of the watermark can be simply proven by applying the input sequence of the watermark and observing the output sequence. If the observed output sequence matches the output sequence of the watermark, the existence of the watermark is necessarily proven.

Regular sequential functions are generally represented by complex and highly optimized automata, developed in both stand-alone and embedded processors. In order to maximally exploit the advantages of a particular technology, there is little room for overhead, in the form of both additional circuits and/or signals. For this reason, the proposed algorithms operate in both active and passive synthesis regimes, and they are designed to prevent excessive implementation overhead for a specified level of detection confidence.

This paper is organized as follows. A formulation of the problem is presented in Section II. Section III outlines the process of modifying the inner structure of regular sequential functions to add the watermark. Detection techniques are presented in Section IV, and examples are given in Section V.

II. General Problem Formulation

In its most general form, a sequential function transforms input sequences into output sequences. Regular sequential functions are functions such that at any stage the output symbol depends only on the sequence of input symbols that have been already received. Any regular sequential function operating on finite input/output sets can be specified by means of an FSM.

An FSM is a discrete dynamical system translating sequences of input vectors into sequences of output vectors, and it is generally represented by state transition graphs (STG's) and state transition tables (STT's). An STG is a graph whose nodes represent the states of the FSM and whose edges determine the input/output conditions for a state-to-state transition. By convention, an edge is labeled by the input/output pair causing the transition.

In real-world sequential designs, although not explicitly specified using STG's and STT's, FSM's appear in different forms. For example, case statements in VHDL and Verilog HDL are represented as FSM's using an STG or STT by HDL compilers. FSM's also appear in embedded software, especially to define the device drivers and interface protocols. In large sequential designs, usually several such small FSM's exist that can be used to watermark the entire design. By watermarking all or a selected subset of these FSM's, tampering resilience can be reached while ensuring the method's feasibility.

The essence of the proposed technique is to find an unused input/output symbol sequence and use it as the watermark. This task can be performed by using the STG representation of the regular sequential function. By visiting every state and finding the unused input/output symbol pairs, one can determine the candidate subset of such symbol pairs at each state in the FSM.

After calculating the required input/output symbol sequence length that satisfies given uniqueness constraints, i.e., constraints on P_u, one can generate a sequence by selecting enough input/output symbol pairs. If the found input/output symbol pairs are not sufficient, then one can create extra ones by augmenting the input and/or output alphabets. The estimation of P_u and the derivation of the length of the input/output symbol sequence will be explained in full detail in Section III.

Last, by connecting the states, one can generate a trace in the FSM. Some selections of input/output symbol sequences and the states may generate large FSM's. In Section III techniques are

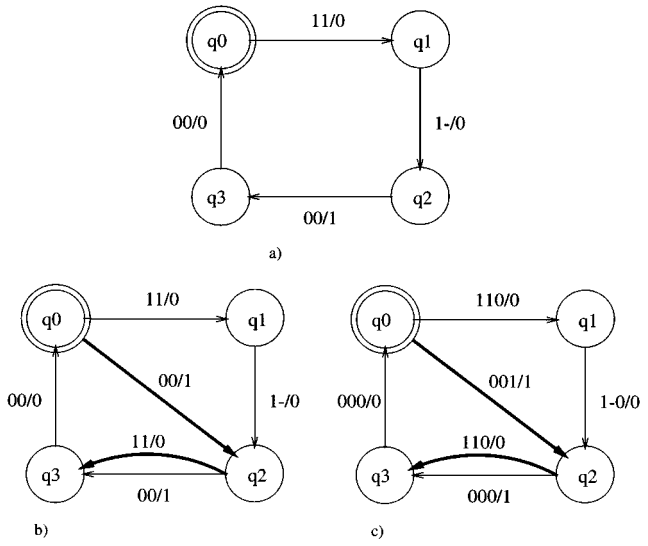

Fig. 1. An example of two possible ways of watermarking an FSM: (a) original FSM, (b) adding transitions, and (c) augmenting input and adding transitions.

proposed to prevent this occurrence by minimizing the FSM's overhead.

To capture the essence of the proposed techniques, consider the example of Fig. 1. The original FSM is depicted in Fig. 1(a) in terms of its STG. The FSM has two input bits and one output bit. Assume one has decided that a watermark of length 2 is satisfactory, and suppose the proposed watermark is represented by input/output sequence ((00,1)(11,0)). Fig. 1(b) illustrates the new FSM obtained after augmentation and state selection.

Assume that the input/output pairs available are not satisfactory. Then, in this case, the number of inputs is first incremented by one (for illustrative purposes). Two extra transition relations can hence be added. The resulting FSM is depicted in Fig. 1(c).

In the remainder of this paper, we will restrict ourselves to deterministic FSM's, using the same notation of [13] and [14].

Definition 1: Let an FSM be a tuple $M = (\Sigma, \Delta, Q, q_0, \delta, \lambda)$, where Σ and Δ are respectively the input and output alphabets, Q is a finite set of states, $q_0 \in Q$ is the initial state, $\delta(q, a) : Q \times \Sigma \rightarrow Q \bigcup \{\phi\}$ is the transition relation, and $\lambda(q, a) : Q \times \Sigma \rightarrow Q \bigcup \{\epsilon\}$ is the output relation.

$q \in Q$, $a \in \Sigma$, and $b \in \Delta$ refer to a state, an input, and an output, respectively. ϕ denotes an unspecified next state, while ϵ is an unspecified output. An FSM can be identified by the mapping of all its input and output sequences, or *IO mapping*.

Definition 2: An IO mapping is defined to be the sequence of input/output pairs $((a_1, b_1), (a_2, b_2), \ldots, (a_k, b_k)) \in (\Sigma \times (\Delta \bigcup \{\epsilon\}))^k$ specifying the output sequence of the FSM for a given input sequence.

Let us define Σ^* and Δ^* as the sets of all strings in Σ and in Δ, respectively. Let $s = (a_1, \ldots, a_k) \in \Sigma^*$ be an arbitrary input sequence, and let $d = (b_1, \ldots, b_k) \in \Delta^*$ be an output sequence. Moreover, define $\lambda(q, s)$ to be the output symbol of the FSM and $\delta(q, s)$ its state when s has been applied in state q. String s is said to be contained in M iff a state reached by applying s to state q_0 is still in M, i.e., iff $\delta(q_0, s) \in Q$.

Completely specified FSM's (CSFSM's) contain every element of set Σ^*, i.e., every input sequence in Σ^* results in a unique output sequence in Δ^*. An incompletely specified FSM (ISFSM) is one in which there exist some transition relations with unspecified destination and/or output, i.e., there exist a set of input sequences for which no output is specified. Call $I_u \subset \Sigma^*$ such a set. Conversely, there exist a set of output sequences that can be produced only by unspecified input sequences. Call $O_u \subset \Delta^*$ such a set. The problem of minimizing the number of states in CSFSM's can be solved in polynomial time [15]. For ISFSM's, the problem is known to be NP-complete [16]. Algorithms for reducing such machines are proposed in [13]–[15].

Let $M' = (\Sigma', \Delta', Q', q_0', \delta', \lambda')$ be an ISFSM and $\mathcal{P}_{M'}$ be the set of all possible completely specified implementations of M'. Thus, for each $p \in \mathcal{P}_{M'}$, every element of I_u and O_u is eventually associated to an element of Δ^* and Σ^*, respectively. Let us select an arbitrary sequence $s_\sigma \subset I_u$ and the corresponding output sequence $d_\sigma \in \Delta^*$. Let tuple $\sigma = \{s_\sigma, d_\sigma\}$, call it *IO signature*.

Consider first an active watermarking regime. The problem of synthesizing a watermark for an ISFSM M' is equivalent to that of finding a minimum sized machine M'', whose specified IO mapping has been augmented by an IO signature σ on specification of M'. It is also required that a robustness constraint specified as $\overline{P_m}$ and $\overline{P_u}$ be satisfied. The problem is formulated as follows.

Problem 1: Minimize size of M'', such that

$$P_m \leq \overline{P_m}, \quad P_u \leq \overline{P_u}. \tag{1}$$

where $\overline{P_m}$ and $\overline{P_u}$ are constraints on the watermark robustness. Note that the size is measured in terms of added states and logic.

Problem 1 can be partitioned into two tasks. The first task consists of computing the size of IO signature σ so as to satisfy the constraints on the confidence. The second task is that of finding the actual IO signature so as to minimize the overhead of M''. The IO signature must be generated with some degree of randomness to ensure that, using the same algorithm, one cannot generate an identical code. The randomized algorithm is controlled by key k, which is provided by the user to control the generation of the IO signature and of the sequence of states activated by it. k is used to select from n best state sequences and IO signatures. In this case, the minimality of the overhead might not be guaranteed.

In case keeping the IO signature secret were not possible, then one of the following approaches could be used. The authentication of the generated IO signature can be achieved by registering the key of a specific design in a third-party data base, similarly as in copyright and trademark registration.

An alternative solution is that of explicitly creating an IO signature based on the method proposed in [11]. The user specifies a string that is converted into a number by standard one-way hash function like MD5. In this manner, one can guarantee that there will be no two identical IO signatures generated by two different strings, and it is computationally intractable to obtain the string from the IO signature. Using this signature, one can find a state sequence that minimizes the overhead, even though an absolute minimum cannot be guaranteed.

Synthesizing watermarks in CSFSM's requires first that the machine be translated onto a ISFSM. This can be accomplished by extending the input and/or output alphabets Σ and Δ. The resulting machine is then handled by solving Problem 1. Hence, the procedure can be seen as a preprocessing step to a general watermark synthesis step.

A passive watermarking scheme consists of generating signature σ from a given ISFSM without modifying the machine itself. The process consists of first minimizing the FSM using, for example, the techniques proposed in [13], thus synthesizing a CSFSM. Then, a subset of all the sections of the nonspecified IO mapping is designated as a IO signature. Randomization of the signature, controlled by key k, is used to select unspecified IO sequences. Hence, the probability of accidentally synthesizing the same watermark is bounded by the degrees of freedom of the algorithm and/or by its level of randomization.

III. IO Signature Generation

In this section, a solution to Problem 1 is proposed. At least two approaches exist to the generation of an IO signature σ. The first involves the generation of new transition relations in the FSM's STG or STT, while the second calls for the augmentation of Σ, Δ, or Q. All these modifications are likely to but do not necessarily increase the size of the machine.

Let $q' \in Q'$ denote a state in an ISFSM M', and let q'_0 be its reset state. Let $I_u^{(q')}$ be the set of all the input configurations in q' for which no next state is specified, and call such configurations *free*. Define U' to be the set of all the states with incompletely specified transition relations, i.e., $U' = \{q' \in Q' \mid |I_u^{(q')}| > 0\}$. The total number of free input configurations n is bounded as follows:

$$n \leq n_{\max} = \sum_{q' \in Q'} \left|I_u^{(q')}\right|. \tag{2}$$

Every state $q' \in U'$ must necessarily be reachable $|I_u^{(q')}|$ times, using each time one of the remaining free input configurations in $I_u^{(q')}$. Suppose that a sequence x exists of all the visited states, and call s the input sequence that forces x. The resulting output sequence d of length n will be one of $[2^{|\Delta|}]^n$ possible implementations. Hence, the odds that an identical sequence will be produced by M is

$$P_u = \frac{1}{\left[2^{|\Delta|}\right]^n - 1}. \tag{3}$$

The second term of the denominator is given by the fact that one such sequence will result from the given input sequence in the CSFSM in $\mathcal{P}_{M'}$. By setting $P_u \leq \overline{P_u}$ and solving (3) with respect to n, one obtains

$$n \geq n_{\min} = \frac{1}{|\Delta|} \log_2 \left|1 + \frac{1}{\overline{P_u}}\right|. \tag{4}$$

In some cases, it is not possible to satisfy both (2) and (4) to meet specification (1), i.e., $n_{\min} > n_{\max}$. Hence, (1) must be relaxed and/or $n_{\max}$ must be increased.

Suppose constraints (2) and (4) were satisfied; then an output sequence $d_\sigma \in \Delta^*$ and the states that can produce it must be selected. The required output is generated by an n-long sequence of states in U'. The sequence can be seen as a path

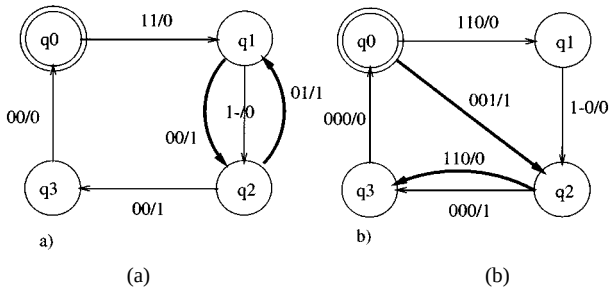

Fig. 2. Two possible paths p_σ for a given U': (a) path based on minimum visited states criterion and (b) path based on maximum remaining free configurations.

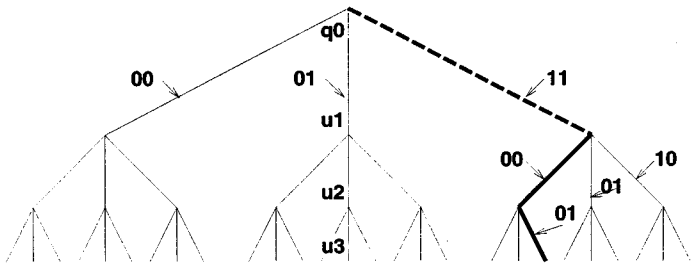

Fig. 3. Decision tree to compute s_σ.

$p_\sigma = (q'_0, u'_1, \ldots, u'_{n-1})$ covering a subset of the states in U', with or without repetition. It is assumed, but not necessary, that $q'_0 \in U'$. If this were not the case, a different first state, say, $q''_0 \in U'$, could be selected for p_σ, and input sequence s_σ would need to be augmented by an input sequence s such that $\delta'(q_0, s) = q''_0$. The generation of p_σ does not contribute to the probability of coincidence P_u, but it does determine the impact that state minimization will have on the final machine. The second factor affecting the effectiveness of the optimization is the selection of input sequence s_σ.

For a given output sequence d_σ, an input sequence s_σ is generated in two steps: selection of p_σ and derivation of s_σ. Sequence p_σ represents a path through n of the states in U' from the original STG. Every time a state u' is touched by the path, it loses one of its $|I_u^{(u')}|$ free input configurations. We propose to use an algorithm based on the Euler path search, which can be targeted to minimize the number of visited states and/or to maximize the number of remaining free configurations per state.

As an illustration, consider the ISFSM example given in Section II. For each state, assume there exist three out of four free input configurations. Assume that $n = 2$; then two possible paths p_σ are shown in Fig. 2(a) and (b). In the example of Fig. 2(a), the number of inputs was unchanged, while in Fig. 2(b) it was incremented by one. Consider the example of Fig. 2(a). Path p_σ, represented in bold, is selected by maximizing the number of remaining free configurations per state. Note that the path may begin in a state other than the reset state q_0. In this case, one must additionally find the input sequence leading to p_σ's initial state.

Once p_σ for Fig. 2(a) has been selected, input sequence s_σ is derived from a path on a decision tree rooted in q_0 and whose leaves correspond to state u'_{n-1}. The solid bold line in Fig. 3 represents p_σ, while the dotted line shows the path needed to reach p_σ's initial state. At each level i, exactly $|I_u^{(u'_i)}| < |\Sigma|$ branches

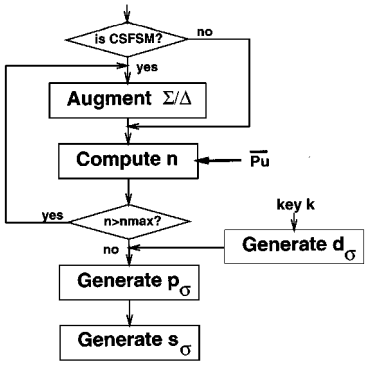

Fig. 4. Active FSM watermarking.

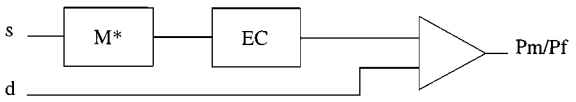

Fig. 5. Detection of signature under some tampering.

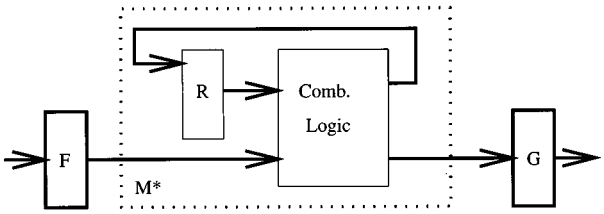

Fig. 6. Tampering based on shuffling and augmentation of I/O bits.

exist. Each branch represents the decision of using a certain free input configuration at a given state. There exists $\Pi_{i=1}^{n}|I_u^{(u_i')}|$ possible paths connecting the root state q_0 to u_{n-1}'. One or more of these paths is associated with the smallest CSFSM $M \equiv M'$. The problem of finding such a path is NP-complete since in the best case the machine associated with one path must be synthesized, which in itself is an NP-complete problem. As an illustration, if the path represented in bold in Fig. 3 is used for ISFSM M', the resulting IO signature is $\{s_\sigma, d_\sigma\} = \{(1,1,0,0,0,1);(0,1,1)\}$.

Several alternatives are proposed for the generation of the input sequence s_σ to minimize overhead. The first method consists of performing an exhaustive search of the decision tree. For each path a CSFSM is synthesized and the smallest machine is selected. The second method is a Monte Carlo approach, in which a set of input sequences are selected at random from all the feasible ones. The CSFSM's corresponding to such sequences are generated and the smallest one is selected. The third method is based on a branch-and-bound search. At each level of the tree an estimate is computed for the machine associated with each subtree underlying any decision. Such an estimate is computed using a Monte Carlo approach. All the subtrees with higher estimates are pruned, while the surviving trees are explored into the next level, i.e., the next state of p_σ. The search stops at the leaves. The complete algorithm for active watermarking in FSM's, shown in Fig. 4, is described as follows.

1) If the FSM is CSFSM, then augment Σ.
2) Compute the minimum size of s_σ, $n_{\min}$, from $\overline{P_u}$.
3) If $n_{\min} > n_{\max}$, then augment Σ or Δ.
4) Using k, randomly generate new output sequence $d_\sigma \in \Delta^*$.
5) Compute path p_σ.
6) Compute input sequence s_σ.

As a by-product of Step 6), the FSM is synthesized. A passive watermarking scheme is applied to ISFSM's only. The method assumes that randomization can be introduced by the FSM synthesis. It consists of converting the original ISFSM onto a CSFSM using a given optimization criterion. Then, an IO signature is selected at random from all the possible ones available. The only way to synthesize a CSFSM from the original ISFSM, which contains an identical IO signature, is to use the same synthesis engine with an identical set of parameters and optimization criteria. Hence, P_u can be derived in this case as the inverse of all possible machines that can be generated from an ISFSM of a certain size and structure with the given engine.

IV. Watermark Detection

In the previous sections, we have proposed techniques to generate an IO signature $\sigma = \{s_\sigma, d_\sigma\}$ and to embed it in the machine. Detecting σ entails applying input sequence s_σ to the machine and observing the output sequence d; see Fig. 5. If no tampering has occurred, then necessarily $d = d_\sigma$ and $P_m = 0$, i.e., no misses are possible. To properly analyze the effects of tampering, let us consider the following scenarios.

1) Specifications on the IO mapping of the original machine are known.
2) IO mapping of the original machine is not known but the STG of the modified machine is known.
3) No STG is known.

In case 1), infringement cannot be prevented, since the aggressor can resynthesize the FSM from specifications using techniques proposed, e.g., in [14].

In case 2), the aggressor may either a) modify state transition relations, i.e., changing the output or next state associated with a transition relation, or b) apply the techniques proposed in this paper to watermark CSFSM's. In both cases, part or the totality of the watermark will be unchanged, but it may be corrupted locally. Tampering a) may in fact result in a change in the functionality of the machine, and it is therefore counterproductive. Tampering b) will only result in literal swaps and deletions within pairs of reset states, similar to gene deletion within DNA sequences.

To combat tampering b), we propose an approach based on the concept of *genome search*. Such an approach was successfully used in topological and netlist watermarking [6], [7]. The method is essentially a selective pattern matching. It is assumed for simplicity that the output d_σ is a chain of sequences all rooted in a single reset state q_0. This restriction is, however, not necessary as multiple reset states can be used. Suppose the IO signature is

$$p_\sigma = (q_0, q_2, q_1, q_0, q_3, q_4, q_0, q_1, q_2, q_0)$$
$$\{s_\sigma, d_\sigma\} = \{(01, 01, 00, 10, 01, 00, 11, 10, 01); (0, 1, 1, 0, 1, 0, 1, 1, 1)\}.$$

TABLE I
IWLS 93 FSM BENCHMARKS. THE NUMBER OF STATES AND THE NUMBER OF I/O PINS REFER TO THE ORIGINAL FSM, WHILE I/O CHG REFERS TO THE MODIFIED FSM. OVERHEAD IS THE EXTRA AREA OF THE MODIFIED FSM.

circuit	# states	# I/O	# I/O chg.	n_{min}	orig. FSM		Monte Carlo		$\overline{P_u}$	Overhead
					area	CPU	area	CPU		
s27	6	4/1	1/3	9	632	0s	1.53k	0.1s	1.4×10^{-11}	143%
bbara	10	4/2	1/1	10	1.16k	0.1s	2.01k	0.1s	9.3×10^{-10}	73%
dk14	7	3/5	1/0	7	1.48k	0.1s	1.84k	0.1s	2.9×10^{-11}	24%
styr	30	9/10	1/0	4	8.6k	0.1s	10.69k	0.1s	9.1×10^{-13}	22%
bbsse	16	7/7	1/0	10	2.28k	0s	2.62k	0.1s	2.9×10^{-11}	6.3%
cse	16	7/7	1/0	5	3.84k	0.1s	4.08k	0.1s	2.9×10^{-11}	6%
sse	16	7/7	0/0	3	2.29k	0s	2.43k	0.1s	4.7×10^{-7}	5.9%
ex1	20	9/19	0/0	4	5.37k	0.1s	5.55k	0.1s	1.3×10^{-23}	3.2%
ex1	20	9/19	0/0	2	5.37k	0.1s	5.40k	0.1s	3.6×10^{-12}	0.6%
viterbi	68	15/59	1/0	2	13.49k	1.5s	13.61k	15s	3.0×10^{-36}	0.8%
dec	56	16/23	1/0	2	14.75k	0.5s	14.78k	5s	1.4×10^{-14}	0.2%
scf	121	27/56	0/0	2	20.97k	3.4s	21.02k	34s	1.9×10^{-34}	0.2%

Suppose that tampering has removed or corrupted the median section of d_σ, i.e., $(0,1,0)$; then the sections of the IO signature that are still intact can be matched to σ using the `genome_search` algorithm described in detail in [6]. The algorithm returns an estimate of the probability that the design contains in fact watermark σ. Note that by construction, it is known when the reset state is reached. Hence, the boundary symbols or *operons* of each "gene" are known. Also note that if this or any other error-correction algorithm is used, then our estimation of P_u is an upper bound on the true value, i.e., it is an optimistic estimate. In this case, changes to the way P_u is estimated should be applied based on the details of the algorithm. An alternative method is that of using correction schemes such as cyclic redundancy check to detect and correct corrupted subsequences.

Last, consider case 3). Let us analyze the possible attempts to remove the watermark using netlist manipulations. Obviously, it is not possible to foresee all possible tampering techniques. Instead, we will analyze those that are more likely to be performed under following assumptions.

Assumption 1: A netlist or a structural HDL description is available for tampering.

Assumption 2: All input and output pins are well documented, and extra I/O pins (if any) used for watermarking are introduced as extra test pins and/or signal pins.

In [11], it has been proven that generating an STG from a given netlist is an NP-complete problem. For medium- and large-scale FSM's, it is unlikely that the STG can be obtained from its netlist. Therefore, if the netlist is obtained by reverse engineering, the aggressor has no other options but to perform one of the following modifications to remove or hide the watermark: a) embed the FSM into a bigger one, b) delete some of the circuitry related to the test inputs, or c) add dummy I/O bits and/or shuffle the bit order using unknown mapping functions.

In scenario a), the aggressor tries to hide the watermark under a wrap to mask the original IP from input/output probing. The watermark is still intact but it may not be easily observable, if at all possible. In this case, the detection technique proposed earlier cannot be exploited. However, simulation or on-chip measurements can be used to logically insulate the original IP from the wrap.

In scenario b), by knowing that the watermark should be related to the extra test pins, the aggressor might try to remove the registers and circuitry related to those inputs. In this case, the attempt would damage the original behavior because the IO signature is an integral part of the FSM. Therefore, this attempt shall not be successful.

In scenario c), the aggressor adds new dummy input and/or output bits and dummy circuitry to the FSM as depicted in Fig. 6. In this case, IP forensic can use the following exhaustive method. Let us assume that there were n input bits and m output bits in the original watermarked FSM. Moreover, assume that d_n and d_m extra bits have been added. Then, one needs to apply the input sequence to each possible subset of n bits of the $n+d_n$ inputs. The output is observed to reconstruct the correct sequence. Although it is time consuming, it is guaranteed that the IO mapping can be found exactly, since the watermark is intact.

V. RESULTS

In our experiments, we have used FSM's from the IWLS93 benchmark set. The tools were implemented in C/C++ and run under UNIX and Linux operating systems. Watermarking was performed on ISFSM's as well as CSFSM's. Constraint $\overline{P_u}$ was selected so as to require, in some cases, expansion of Σ and/or Δ. The increase in the number of states $|Q|$ and input/output bits $|\Sigma|$ is expressed by the area estimates. The estimates are based on technology mapping obtained with SIS[17] using the MSU script. Table I lists all relevant experimental data and specifications on the robustness of the watermark. For the FSM minimization stage in the algorithm of Fig. 4, the tools STAMINA and NOVA [13] were used. The area results are based on the actual circuit implementation after technology mapping obtained via SIS and related to the number of gates.

As expected, larger FSM's require less overhead for comparable robustness. Note, as shown in benchmark `ex1`, that overhead can be traded for smaller values of $\overline{P_u}$. These overhead results are comparable to those obtained in [11]. The overhead of benchmark `s27` was extremely high due to the increase of the output alphabet. Such expansion was, however, necessary to boost the watermark's confidence.

Exhaustive search could be performed only in `sse` due to the extreme computational complexity of the method. The CPU time in this case was 1.0 s for an area of 2.33-k gates. For the other circuits, an estimate of a lower bound of the time required by the search can be computed. Such time estimates are derived multiplying the time required by one minimization with the minimum number of free configurations, i.e., $2^{|I_u^{(\min)}|} n_{\min}$, where $|I_u^{(\min)}| = \min_{q' \in U'} |I_u^{(q')}|$.

In the Monte Carlo approach, a maximum of ten input sequences s_σ was explored. Alternatively, one could select such an upper bound based on some estimate or measurement of the standard deviation of the minimized machine's size. Currently, the authors are working on an efficient implementation of the branch and bound approach.

VI. Conclusion

A watermark-based scheme has been proposed to protect the intellectual property content of regular sequential functions operating on finite input/output sets. By modeling such functions as finite state machines and exploiting some unutilized input vectors, modifications were introduced so as to trigger a specific response with known input excitations. It was shown how the odds of reproducing identical behavior can be made arbitrarily small. It was also demonstrated how machines, which have been infringed upon, are effectively detected. Industrial examples illustrate the effectiveness of the approach.

References

[1] Virtual Socket Interface Alliance. [Online]. Available: http://www.vsi.org

[2] M. D. Swanson, B. Zhu, and A. H. Tewfik, "Transparent Robust Image Watermarking," in *Proc. IEEE Int. Conf. Image Processing*, vol. 3, Sept. 1996, pp. 211–214.

[3] L. Boney, A. H. Tewfik, and K. N. Hamdy, "Digital Watermarks for Audio Signals," in *Proc. IEEE Int. Conf. Multimedia Computing and Systems*, June 1996, pp. 473–480.

[4] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "FPGA Fingerprinting Techniques for Protecting Intellectual Property," in *Proc. IEEE Custom Integrated Circuit Conf.*, May 1998, pp. 299–302.

[5] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "Robust FPGA Intellectual Property Protection through Multiple Small Watermarks," in *Proc. IEEE/ACM Design Automation Conf.*, June 1999, pp. 831–836.

[6] E. Charbon, "Hierarchical Watermarking in IC Design," in *Proc. IEEE Custom Integrated Circuit Conf.*, May 1998, pp. 295–298.

[7] E. Charbon and I. Torunoglu, "Watermarking Layout Topologies," in *Proc. IEEE Asia South-Pacific Design Automation Conf.*, Jan. 1999, pp. 213–216.

[8] A. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe, "Watermarking Techniques for Intellectual Property Protection," in *Proc. IEEE/ACM Design Automation Conf.*, June 1998, pp. 776–781.

[9] D. Kirovski, Y. Y. Hwang, M. Potkonjak, and J. Cong, "Intellectual Property Protection by Watermarking Combinational Logic Synthesis Solutions," in *Proc. IEEE Int. Conf. Computer Aided Design*, Nov. 1998, pp. 194–198.

[10] I. Torunoglu and E. Charbon, "Watermarking-Based Copyright Protection of Sequential Functions," in *Proc. IEEE Custom Integrated Circuit Conf.*, May 1999, pp. 35–38.

[11] A. L. Oliveira, "Robust Techniques for Watermarking Sequential Circuit Designs," in *Proc. IEEE/ACM Design Automation Conf.*, June 1999, pp. 837–842.

[12] E. Charbon and I. Torunoglu, "Copyright Protection of Designs Based on Multi Source IPs," in *Proc. IEEE Int. Conf. Computer Aided Design*, Nov. 1999, pp. 591–595.

[13] T. Villa, T. Kam, R. Brayton, and A. Sangiovanni-Vincentelli, *Synthesis of Finite State Machines: Logic Optimization*. Boston, MA: Kluwer Academic, 1997.

[14] J. M. Pena and A. L. Oliveira, "A New Algorithm for the Reduction of Incompletely Specified Finite State Machines," in *Proc. IEEE Int. Conf. Computer Aided Design*, Nov. 1998, pp. 482–489.

[15] G. De Micheli, *Synthesis and Optimization of Digital Circuits*. New York: McGraw-Hill, 1994.

[16] C. F. Pfleeger, "State Reduction in Completely Specified Finite State Machines," *IEEE Trans. Comput.*, vol. C-22, pp. 1099–1102, 1973.

[17] E. M. Sentovich, K. J. Singh, C. Moon, H. Savoj, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Sequential Circuit Design Using Synthesis and Optimization," in *Proc. IEEE Int. Conf. Computer Design*, Oct. 1992, pp. 328–333.

Ilhami Torunoglu received the M.S. degree in electrical and electronic engineering from the Middle East Technical University, Ankara, Turkey.

In his graduate studies, he has specialized in physical design automation tools such as editors, compactors, and placers. He has more than six years of hands-on experience in developing tools and algorithms for computer-aided design. In addition, he has more than five years of experience in VLSI chip design. He has worked for Cadence Design Systems, TUBITAK TAEGE (National Electronic Research Center), and was instrumental in the establishment of the first fabless design center in Turkey. In recent years, he has been a Member of Consulting Staff at Cadence. In this capacity, he was responsible for leading a development team of engineers. He was responsible for technology and product development of the Virtuoso compactor and layout synthesis products. He is the author of ten publications. His current interest areas are full-custom layout automation, IP protection, and embedded systems.

Edoardo Charbon (S'90–M'92) received the diploma in electrical engineering from the Swiss Federal Institute of Technology (ETH), Zurich, in 1988, the M.S. degree in electrical and computer engineering from the University of California at San Diego in 1991, and the Ph.D. degree from the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in 1995.

Between 1988 and 1989, he worked at the Department of Electrical Engineering, ETH, where he designed CMOS A/D converters for integrated sensor applications. In 1989, he visited the Department of Electrical Engineering of the University of Waterloo, Canada, where he was involved in the design and fabrication of ultra-low-noise, nanotesla magnetic sensors. At Berkeley, he worked on performance-directed, constraint-based analog and mixed-signal physical design automation and accelerated substrate extraction techniques. Since 1995, he has been with Cadence Design Systems, where he is leading the development effort on constraint management in the physical design group. He is also the Project Leader of Cadence's first methodology for intellectual property protection. He has published more than 40 articles in technical journals and conference proceedings and a book, and has been consulting with Texas Instruments and Hewlett-Packard. His research interests include CAD for radio-frequency IC's, methodologies for intellectual property protection, substrate modeling and characterization, superconducting parasitic analysis, and micromachined sensor design.

Dr. Charbon has been a Guest Editor of the IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.