A Robust FSM Watermarking Scheme for IP Protection of Sequential Circuit Design

Aijiao Cui, *Member, IEEE*, Chip-Hong Chang, *Senior Member, IEEE*, Sofiène Tahar, *Senior Member, IEEE, and* Amr T. Abdel-Hamid, *Member, IEEE*

***Abstract*—Finite state machines (FSMs) are the backbone of sequential circuit design. In this paper, a new FSM watermarking scheme is proposed by making the authorship information a non-redundant property of the FSM. To overcome the vulnerability to state removal attack and minimize the design overhead, the watermark bits are seamlessly interwoven into the outputs of the existing and free transitions of state transition graph (STG). Unlike other transition-based STG watermarking, pseudo input variables have been reduced and made functionally indiscernible by the notion of reserved free literal. The assignment of reserved literals is exploited to minimize the overhead of watermarking and make the watermarked FSM fallible upon removal of any pseudo input variable. A direct and convenient detection scheme is also proposed to allow the watermark on the FSM to be publicly detectable. Experimental results on the watermarked circuits from the ISCAS'89 and IWLS'93 benchmark sets show lower or acceptably low overheads with higher tamper resilience and stronger authorship proof in comparison with related watermarking schemes for sequential functions.**

***Index Terms*— IP Protection, IP Watermarking, Sequential Design, Finite State Machine, State Transition Graph.**

I. INTRODUCTION

As reuse-based design methodology has taken hold, the VLSI design industry is confronted with the increasing threat of intellectual property (IP) infringement. IP providers are in pressing need of a convenient means to track the illegal redistribution of the sold IPs. An active approach to protect a VLSI design against IP infringement is by embedding a signature that can only be uniquely generated by the IP author into the design during the process of its creation. When a forgery is suspected, the signature can be recovered from the misappropriated IP to serve as undeniable authorship proof in front of a court. Such a copyright protection method is widely known as *watermarking*. It is cheaper and more effective than patenting or copyrighting by law to deter IP piracy [1].

Unlike the digital content in the media industry, a VLSI IP is developed in several levels of design abstraction with the help of many sophisticated electronic design automation tools. Each level of design abstraction involves solving some NP-complete optimization problems to satisfy a set of design constraints. In the regime of *constraint-based watermarking*, the signature to be imprinted is converted into a set of extra constraints to be extraneously satisfied by the watermarked design [2]. The watermark embedded at a higher level of design abstraction must survive the posterior optimizations so that the same IP distributed at all lower abstraction levels are protected. From the authorship verification perspective, IP watermarking can be classified into static watermarking and dynamic watermarking [3]. In the watermark detection phase, static watermarking [4]-[8] requires the downstream design to be reverse engineered to the level where the watermark is embedded to show the additional constraints generated by the author's signature are satisfied. Reverse engineering is expensive and intrusive as some critical design data used to produce the watermarked IP may be exposed in this process. On the other hand, dynamic watermarking [9]-[17] enables the embedded information to be detected from the output without reverse engineering by running the protected design with a specific code sequence. Dynamic watermarking is typically performed in the state transition graph (STG) of finite state machine (FSM) [11]-[14], in the architectural level of digital signal processors (DSP) [9], [10] or at the design-for-testability (DfT) stage [15]-[17]. FSM watermarking embeds the signature at a higher (behavioral/RT) level of design abstraction whereas the latter normally embeds the signature after logic synthesis. Embedding the watermark at the behavioral level has the advantage that it is harder for the attacker to erase the watermark in the downstream design by simple redundancy removal or logic manipulation, but it is also challenging to keep the overhead of watermarked design low.

In this paper, a new dynamic watermarking scheme is proposed. The watermark is embedded in the state transitions of FSM at the behavioral level. As an FSM design is usually specified by an STG or other behavioral descriptions that can be easily translated into STG, the watermark is embedded into the STG of any size and remains a property of FSM after the watermarked design is synthesized and optimized into circuit netlist. The authorship can be directly verified even after the downstream integrated circuit design processes by running the watermarked FSM with a specific code sequence. Unlike [12], our watermark verification is simple and efficient even for large

Manuscript received Dec. 28, 2009; Revised July 12, 2010; Accepted Nov. 11, 2010.

A. Cui is with the Department of Electronic and Information Engineering, Harbin Institute of Technology Shenzhen Graduate School, GuangDong Province, P. R. China 518055 (e-mail: cuiaj@hitsz.edu.cn).

C. H. Chang is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: echchang@ntu.edu.sg).

Sofiène Tahar is with the Department of Electrical and Computer Engineering, Concordia University, Montreal, Quebec, Canada (e-mail: tahar@encs.concordia.ca).

A. T. Abdel-Hamid is with the German University in Cairo, Egypt (e-mail: amtalaat@gmail.com).

designs. On the other hand, as extracting the STG from a gate level netlist is computationally impractical for large circuits [11], there are limited options for an attacker to remove or hide the watermark from the watermarked design netlist or netlist obtained by reverse engineering its downstream design [13]. The proposed watermarking scheme is robust against state reduction attacks. It is different from other transition-based embedding methods [13], [14] in that it has lower embedding overhead and has overcome the vulnerability of auxiliary inputs which are inevitably introduced if the embedding capacity is limited, especially for completely specified FSM. The weaknesses of existing FSM watermarking scheme to be overcome in this paper are discussed in the next section. Currently there is no easy way to publicly detect the existence of watermark, once the FSM is integrated into a chip and packaged [11]-[14]. Since the test signals can be traced after the chip is packaged and the scan path provides controlled accesses to all internal states and combinational circuits of the watermarked IP, this paper also proposes an alternative approach to allow the authorship proof of watermarked FSM to be verified off chip by making it a part of the test kernel. The proposed watermarking scheme thus makes the authorship proof harder to erase and the IP authorship easier to verify.

The rest of the paper is organized as follows. In Section II, we discuss related works. Our new FSM watermarking scheme is presented in Section III. In Section IV, we analyze the resilience of the proposed watermarking method. Section V presents experimental results on benchmark designs. Finally, Section VI concludes the paper.

II. Related Works

The notion of constraint-based watermarking, first proposed by Hong and Potkonjak, [2] has now been widely applied to embed authorship signature into VLSI designs developed at different design abstraction levels, such as architectural level [9], [10], combinational logic synthesis level [4]-[7] and physical placement and routing [8]. At behavior level, STG representation makes watermarking FSMs in industrial designs promising as efficient sequential logic synthesis tools and optimization methods are available to lower the cost of embedding and detection of watermark. FSM watermarking has the advantage that the IP author signature can be lucidly recovered by applying a verification code sequence. As the STG is in general exponentially larger than the circuit description itself [12], it is computationally impractical to analyze the circuit to extract the STG. Such scheme therefore has high resilience against tampering at lower abstraction levels.

An FSM is characterized by a set of internal states and transitions between them. Approaches to FSM watermarking can be classified based on whether the authorship information is embedded in the states [11], [12] or on the transitions [13], [14]. In [12], 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 upon removal of all redundant states by the application of a state minimization program [18]-[20]. Watermarking on the states of FSM is thus vulnerable to state optimization attacks. Two possible ways to verify the presence of a watermark are provided in [12]. The implicit BDD-based enumeration method is too slow for large circuits. The ATPG-based method requires the solution of an NP-complete problem and is not evident that the verification can be carried out efficiently on large circuits.

The properties of the transitions in FSM can also be explored for watermark embedding. An FSM watermarking scheme was proposed in [13] by inserting redundant transitions into the original STG after the unspecified transitions in the STG are searched and associated with the user-defined input/output sequence. The weakness of this scheme is the monotonous use of only the unspecified transitions with the specified outputs of STG for watermark insertion. The embedding capacity is limited by the number of free input combinations. For FSMs with limited unspecified transitions, the probability of coincidence is high. If the watermark length is increased beyond the available number of unspecified transitions to boost the authorship proof, the overhead aggravates rapidly.

To increase the robustness of FSM watermarking, besides the unspecified transitions, existing transitions are also utilized in an output mapping algorithm to watermark the FSM [14]. This method takes advantage of the original transitions in the STG to lower the overhead of watermarking. The embedding process is fast as no special effort is made to search the states of STG. The watermark bits are embedded 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 automatically watermarked. Otherwise, extra watermarked transition will be added to the STG. When the number of outputs of FSM increases or when the FSM is completely specified, output coincidence of existing transition with the watermark bits becomes rare. The watermarked FSM is susceptible to removal attack if the ratio of augmented transitions to coinciding transitions is high. When only unspecified transitions are watermarked, the scheme becomes as vulnerable as [13]. If no unspecified transitions are available for watermarking, a pseudo input variable is added. This input variable is assigned a fixed logic value of "0" for all existing transitions, and a fixed "1" for the added transitions. This discrimination between the existing transitions and added transitions is conspicuous. Moreover, the addition of new input variables with fixed assignments on all transitions increases the decoder logics and hence the overhead of watermarked FSM significantly. Removal of the pseudo inputs can easily eliminate or corrupt the watermark without affecting the FSM functionality.

In what follow, a more robust technique of transition-based FSM watermarking is proposed to overcome the shortcomings of the above methods. Provision is also made to facilitate the FSM watermark to be readily verified off-chip through the scan chain.

III. Finite State Machine Watermarking

A. *Preliminaries*

A formal definition of an FSM is given in [19] as follows:

Definition 1: An FSM is a tuple $M = (\Sigma, \Delta, Q, s_0, \delta, \lambda)$, where Σ and Δ are finite, non-empty sets of the input and output

alphabets, respectively. Q is a finite, non-empty set of states and $s_0 \in Q$ represents a unique reset state. $\delta(s, X)$: $Q \times \Sigma \rightarrow Q \cup \{\varnothing\}$ is the state transition function and $\lambda(s, X)$: $Q \times \Sigma = \Delta \cup \{\tau\}$ is the output function, where $\varnothing$ denotes an unspecified state and τ denotes an unspecified output.

For $s_i, s_j \in Q$, s_j is said to be the next state of s_i if $\exists X \in \Sigma$ s.t. $s_j = \delta(s_i, X)$. The application of X on s_i also produces an output, $Y = \lambda(s_i, X) \in \Delta$. For an FSM with n input and k output variables, each input alphabet, $X = x_1 x_2 \ldots x_n$, is a string of n bits and each output alphabet, $Y = y_1 y_2 \ldots y_k$, is a string of k bits. Each bit of X and Y, $x_i, y_i \in \{0, 1, -\}$, where "0" and "1" are the binary constants, and "–" denotes a "don't care" value. To avoid unnecessary notational complexity, we use an upper case letter to denote an input or output alphabet in Σ and Δ, a lower case letter to denote an input or output variable in $\{0, 1, -\}$, and $y_{i,j}$ to address the j-th bit of the i-th alphabet, Y_i.

FSMs are usually designed with their state transition graph (STG). An STG, $STG(M) = G(V, E)$, is a labeled directed graph of a machine M of V nodes and E edges. Each symbolic state, $s \in Q$, is represented by a node in V. A state transition t from a source node $S(t)$ to a destination node $D(t)$ is represented by a directed edge, $e_{ij} \in E$, connecting $S(t)$ to $D(t)$. Each edge is tagged with an input/output label, $I(t)/O(t)$, to encapsulate the relations, $D(t) = \delta(S(t), I(t))$ and $O(t) = \lambda(S(t), I(t))$. Thus, a state transition t can be represented by a quadruple $(S(t), D(t), I(t), O(t))$. The input combinations that are absent from all transitions of a source state in an STG are called the *free* (*or unspecified*) *input combinations* of that state, and a transition that can be created from the free input combinations is called an unspecified transition. Unlike [12], as the number of states in an FSM is a dominant factor of the implementation complexity, we modify only the properties of the edge set to synthesize the watermarked design in order to preserve the nodes in $STG(M)$.

In light of dynamic watermarking, the watermark detection process involves the abstraction of an output sequence, $\hat{\mathbf{Y}} = \{\hat{Y}_1, \hat{Y}_2, \cdots, \hat{Y}_N\}$, $\hat{Y}_i \in \Delta$, from the watermarked design $\hat{M}$ by applying a specific input sequence, $\hat{\mathbf{X}} = \{\hat{X}_1, \hat{X}_2, \cdots, \hat{X}_N\}$, $\hat{X}_i \in \Sigma$, on a state, $\hat{s} \in Q$, such that $\hat{\mathbf{Y}} = \lambda(\hat{s}, \hat{\mathbf{X}}) = \lambda\left(\delta\left(\delta\left(\cdots\delta\left(\hat{s}, \hat{X}_1\right)\cdots\right), \hat{X}_{N-1}\right), \hat{X}_N\right)$. The watermark synthesis process requires that the outputs of $\hat{M}$ be compatible with the outputs of M for every input symbol, $X \in \Sigma$, and output mappings of $\hat{M}$ for every input symbol, $\hat{X}_i \in \Sigma \; \forall i = [1, N]$, be dictated by a signature that identifies the ownership of a design. The signature is cryptographically generated with a secret key so that $\hat{\mathbf{Y}} = \lambda(\hat{s}, \hat{\mathbf{X}})$ becomes a unique property of $\hat{M}$.

In [13], [14], the length N of $\hat{\mathbf{X}}$ and $\hat{\mathbf{Y}}$ is equal to m/k, where m is the watermark length and k is the number of output variables of an FSM. Fig. 1(a) shows an example of a STG with three states, S_1, S_2 and S_3. The state transitions are determined by a 1-bit input variable and a 3-bit output variable, i.e., $n = 1$ and $k = 3$. When the scheme in [14] is applied to embed an 8-bit watermark sequence "10101000", three ($m/k = 3$) consecutive transitions will be searched to match the watermark bits with the output bits. If the search starts from S_1, as all transitions from S_1 have no output coinciding with the first three watermark bits of "101", a new transition will be inserted. Since S_1 has no free input combination, a new input variable is introduced. This input variable is assigned to "0" for all existing transitions and "1" for all added transitions, and the bits are underlined in Fig. 1(b). A new transition (S_1, S_2, 11, 101) from S_1 is added with an arbitrarily chosen next state S_2 as indicated by the bold dashed arc in Fig. 1(b). As S_2 has no edge with output bits coinciding with "010", another new transition (S_2, S_3, 01, 010) is added with the randomly selected next state S_3. The existing transition (S_3, S_1, 10, 001), printed bold in Fig. 1(b), has an output matching with the watermark bits "00". So it is reused for watermarking. The watermarked design synthesized by SIS [23] has 640 units of area, 7.2 units of delay and 201.8 units of power. Comparing with the original design with 448, 6 and 178 units of area, delay and power, respectively, the FSM watermarked by [14] incurs 42.9%, 20% and 13.4% overheads in area, delay and power, respectively.

In this example, the output is a 3-bit ($k = 3$) alphabet. The probability of the output of a transition coinciding with the watermark bits is as low as 1/8, which results in only one out of three existing transitions being used for watermarking. When k is larger, it becomes more difficult to make use of existing transitions to reduce the overhead of watermarking due to the low probability of output coincidence. The fixed assignment of the added input variable also increases the design complexity. Moreover, as all output bits are watermarked in consecutive transitions after the starting state on which $\hat{\mathbf{X}}$ is applied, as shown in Fig. 1(c), the watermarked transitions are not well obfuscated, causing the watermarked FSM to be vulnerable.

To overcome these problems, we make $N > m/k$ so that not all bits in $\hat{\mathbf{Y}}$ are watermarked. The locality of the watermark is randomized by a cryptographic one-way function such that any number (from 1 to k) of bits at any output bit from any transition of STG is probable to be watermarked. The general idea can be illustrated using the same STG example in Fig. 1(a). Since $N > 8/3$, it is set to 8. The localities of these 8 watermark bits are randomly generated between $[1, k \times N = 24]$ without replication. Suppose these numbers are $\{9, 13, 2, 10, 20, 23, 17, 4\}$. So, eight transitions will be sought to produce an output sequence that contains the watermark sequence "10101000" at these bit positions in the output. As the 8 watermark bits are dispersed into 8 transitions, the probability of the output of an existing transition coinciding with the watermark bit is as high as 1/2, which results in five existing transitions being reused for watermarking and only one new transition is added, as shown in Fig. 1(d). As the newly added transition is well blent with the existing transitions, when $\hat{\mathbf{X}}$ is applied on the FSM to detect the watermark, it is hard for an attacker to differentiate it from others, as indicated by the bold arrow in Fig. 1(e). To increase the watermark strength and minimize the next state decoder logic of watermarked design, we also capitalize on the extra headroom created by the pseudo input variables and free input combinations of the FSM. In Fig. 1(d), when a new input variable is introduced, it does not need to be fixed and it can remain as *don't care* in the final watermarked design if it is not used for the generation of any new transitions. The synthesized design from Fig. 1(d) has 520, 6.4 and 190.2 units of area, delay and power, respectively. The overheads due to watermarking are

only 16.1% on area, 6.7% on timing and 6.9% on power. The advantage over [14] is discernible.

With these preliminaries, our proposed FSM watermarking algorithm will be elaborated next.

B. *Generation of Watermark and Random Sequence.*

A meaningful text string, *MSG* is first encoded into a binary string and then encrypted by a provable cryptographic algorithm with the secret key K_e of the IP owner. If the length of the encrypted message is too long, a message digest (MD) algorithm can be used to reduce its length. The resultant binary bit vector of length m is the watermark, $W = \{w_i\}_{i=1}^{m}$ and $w_i \in \{0,1\}$.

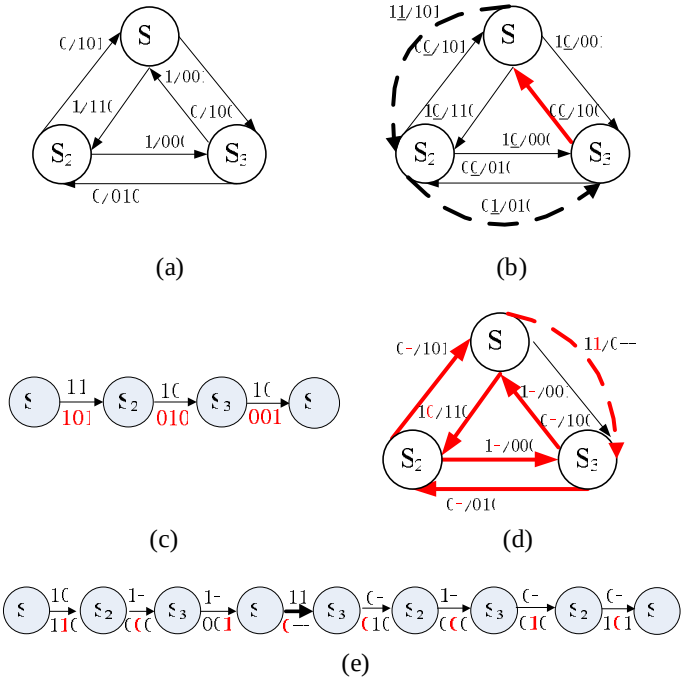

Fig. 1. Watermark embedding on transitions of STG: (a) original STG, (b) watermarked STG by the scheme in [14], (c) excitation of watermarked transitions of STG in (b), (d) watermarked STG by proposed scheme, (e) excitation of watermarked transitions of STG in (d).

A keyed one-way pseudorandom number generator (PNG) is used to generate a sequence, $B = \{b_i\}_{i=1}^{m}$, of m unique integers between 1 and $N \times k$, i.e., $b_i \in [1, N \times k]$ $\forall i = 1, 2, \ldots, m$ and $b_i \neq b_j$ $\forall i \neq j$. The length N of sequence $\hat{\mathbf{X}}$ is determined empirically. The purpose of B is to randomly disperse the m watermark bits into $\hat{\mathbf{Y}}$. If $\exists (i, j)$ $\forall i \in [1, N]$ and $j \in [1, k]$ such that $(i-1)k + j = b_l$, then $\hat{y}_{i,j} = w_l$, where $\hat{y}_{i,j}$ is the j-th bit of $\hat{Y}_i \in \hat{\mathbf{Y}}$. The secure hash algorithm SHA-1 [21] can be used as an MD as well as in a keyed one-way PNG for the generation of these two random sequences, W and B. As it is computationally infeasible to find a collision of this hash function, the possibility that the same group of numbers is generated by coincidence is extremely low without the knowledge of the secret key.

C. *Watermarking Insertion*

The watermark W is inserted into $STG(M)$ by modifying some of its edges without changing the operational behavior of M to find a sequence of N consecutive transitions, $\hat{t}_i = (\hat{s}_i, \hat{s}_{i+1}, \hat{X}_i, \hat{Y}_i)$, $i = 1, 2, \ldots, N$, such that each watermark bit, $w_l \in W$, $l \in [1, m]$, will be randomly mapped to one bit in the sequence, $\hat{\mathbf{Y}} = \hat{Y}_1 \hat{Y}_2 \cdots \hat{Y}_N = \hat{y}_{1,1} \cdots \hat{y}_{1,k} \hat{y}_{2,1} \cdots \hat{y}_{2,k} \cdots \hat{y}_{N,1} \cdots \hat{y}_{N,k}$. The mapping from W to $\hat{\mathbf{Y}}$ is injective but not surjective. The value of each bit $\hat{y}_{i,j}$ in $\hat{\mathbf{Y}}$ can be determined as follows: if $(i-1)k + j = b_l$, then $\hat{y}_{i,j} = w_l$, else $\hat{y}_{i,j} =$ "–", as shown in Fig. 2.

Given an output $\hat{Y}_i$ and a source state $\hat{s}_i$, the destination state $\hat{s}_{i+1}$ of watermarked transition $\hat{t}_i$ will be determined by an output compatibility check. Two bits, x, $y \in \{0, 1, -\}$, are compatible if they are of equal value or one of them has a don't care value, i.e., $x \cap y \neq \varnothing$. This intersection of two ternary variables is defined in Table I. Likewise, two alphabets, X and Y are compatible, denoted by $X \equiv Y$, if none of the elements in $X \cap Y = \{x_i \cap y_i\}$ has a null value.

Generate $\hat{\mathbf{Y}}$ (W, B) {
  $\hat{\mathbf{Y}} = \{\hat{y}_{i,j}\}$, $i \in [1, N]$, $j \in [1, k]$;
  for (i = 1 **to** N) {
    for (j = 1 **to** k) {
      $\hat{y}_{i,j} = -$;
      for (l = 1 **to** m) {
        if $((i-1)k + j = b_l)$ {
          $\hat{y}_{i,j} = w_l$;
          break; } }
    } }
  return $\hat{\mathbf{Y}}$;
}

Fig. 2. Generation of watermarked output sequence.

TABLE I Intersection of two ternary variables

∩	0	1	–
0	0	∅	0
1	∅	1	1
–	0	1	–

Starting with $i = 1$, an arbitrary state, $\hat{s}_1 \in Q$, is selected. Let $T(\hat{s}_i)$ be the set of transitions emanating from a state, $\hat{s}_i$. A set of transitions $C(\hat{s}_i)$ that is output compatible with $\hat{Y}_i$ is sought, i.e., $C(\hat{s}_i) = \{t_i \in T(\hat{s}_i) | O(t_i) \equiv \hat{Y}_1\}$. To avoid entering into a deadlock, transitions terminated at a deadlock state (i.e., state with no fanout) are excluded from $C(\hat{s}_i)$. Four distinct scenarios are considered for the determination of $\hat{t}_i$.

Cases 1: There is only one output compatible transition, $|C(\hat{s}_i)| = 1$, then $\hat{t}_i = C(\hat{s}_i)$ and $\hat{s}_{i+1} = D(\hat{t}_i)$.

Case 2: If more than one output compatible transition are found, i.e., $|C(\hat{s}_i)| > 1$, then a transition from $C(\hat{s}_i)$, with the next state having the highest number of free input combinations, will be selected as $\hat{t}_i$. Its output will be modified to $O(\hat{t}_i) = O(\hat{t}_i) \cap \hat{Y}_i$ and $\hat{s}_{i+1} = D(\hat{t}_i)$.

Case 3: If $|C(\hat{s}_i)| = 0$, then the free input combinations of $\hat{s}_i$ will be considered. Let $F(\hat{s}_i) = \{X \in \Sigma \mid \delta(\hat{s}_i, X) = \varnothing\}$ be the

set of free input combinations of $\hat{s}_i$. For $F(\hat{s}_i) \neq \varnothing$, let $D(\hat{s}_i) = \{\hat{s}_j \in Q \mid \hat{s}_j = D(\hat{t}_i) \; \forall \hat{t}_i \in T(\hat{s}_i)\}$ be the set of all destination states of $\hat{s}_i$. $\hat{s}_{i+1}$ is set to the state with the highest number of free input combinations in $D(\hat{s}_i)$ (excluding the deadlock states) unless $D(\hat{s}_i) = \varnothing$. When $D(\hat{s}_i) = \varnothing$, $\hat{s}_{i+1}$ is set to the state with the highest number of free input combinations in *STG*(*M*). If there exists an edge connecting $\hat{s}_i$ to $\hat{s}_{i+1}$ in *STG*(*M*), a new input/output pair, $I(\hat{t}_i)/O(\hat{t}_i)$, is added for the transition $\hat{t}_i$. Otherwise, a new edge directed from $\hat{s}_i$ to $\hat{s}_{i+1}$ labeled with $I(\hat{t}_i)/O(\hat{t}_i)$ will be created in *STG*(*M*) for $\hat{t}_i$, and $O(\hat{t}_i) = \hat{Y}_i$. The determination of $I(\hat{t}_i)$ will be explained later.

Case 4: If $|C(\hat{s}_i)| = 0$ and $F(\hat{s}_i) = \varnothing$, then a pseudo input variable x_{n+1} needs to be introduced in *M* and the number of input variables *n* is incremented by 1. x_{n+1} is set to an unspecified logic value "*" for all existing transitions. A new edge directed from $\hat{s}_i$ to $\hat{s}_{i+1}$ labeled with $I(\hat{t}_i)/O(\hat{t}_i)$ will be created for $\hat{t}_i$. $\hat{s}_{i+1}$ is set to the state with the highest number of free inputs in $D(\hat{s}_i)$ or in *STG*(*M*) if $D(\hat{s}_i) = \varnothing$, and $O(\hat{t}_i) = \hat{Y}_i$. Both symbols "*" and "–" can assume either a logic "0" or a logic "1" value but there is a subtle difference. "–" is meant for the currently used input combinations whereas "*" can be associated with either the used or free input combinations. A "*" can be construed as a reserved free input literal as its logic state ("0" or "1") will only be defined at the time when some input combinations subsumed by it are freed to become $I(\hat{t}_i)$.

```
Find t̂_i (Q, i, M̂, Ŷ, ŝ_i) {
  if (|C(ŝ_i)| = 1) {                          // case 1
     t̂_i = C(ŝ_i); ŝ_{i+1} = D(t̂_i);
  } else if |C(ŝ_i)|>1 {                       // case 2
     t̂_i = T(arg{max(F(s_j))}) ∀ s_j ∈ D(ŝ_i) and T(s_j) ∈ C(ŝ_i);
     O(t̂_i) = O(t̂_i) ∩ Ŷ_i ; ŝ_{i+1} = D(t̂_i);
  } else {
     if ( F(ŝ_i) ≠ ∅ ) {                       // case 3
        if ( D(ŝ_i) ≠ ∅ ) ŝ_{i+1} = arg{max(F(s_j))} ∀ s_j ∈ D(ŝ_i);
        else ŝ_{i+1} = arg{max(F(s_j))} ∀ s_j ∈ Q;
     } else {                                  // case 4
        Add a pseudo input variable, x_{n+1};
        for (each t̂ ∈ M̂ ) I(t̂)_{n+1} = *;  // set x_{n+1} to *
        n = n +1;
        if ( D(ŝ_i) ≠ ∅ ) ŝ_{i+1} = arg{max(F(s_j))} ∀ s_j ∈ D(ŝ_i);
        else ŝ_{i+1} = arg{max(F(s_j))} ∀ s_j ∈ Q;
     }
     O(t̂_i) = Ŷ_i;
     I(t̂_i) = Find I(t̂_i)(n, M̂, ŝ_i);
  }
  return t̂_i;
}
```

Fig. 3. Determination of watermarked transition.

The pseudo codes for the determination of watermarked transitions are shown in Fig. 3. The input alphabets for the watermarked transitions found in Cases 3 and 4 are determined by the subroutine **Find** shown in Fig. 4.

When there is no existing transition with compatible output, as in Cases 3 and 4, the input alphabet $I(\hat{t}_i)$ for $O(\hat{t}_i) = \lambda(\hat{s}_i, I(\hat{t}_i)) = \hat{Y}_i$ needs to be determined. $I(\hat{t}_i)$ is set to one of the free input combinations of $\hat{s}_i$ if no "*" appears in all the used input combinations of $\hat{s}_i$. Otherwise, an alphabet, $X \in I(t_u)$, $t_u \in T(\hat{s}_i)$, that contains at least one "*" from the set of used input combinations of $\hat{s}_i$ will be split into two. Initially, $I(\hat{t}_i) = X$. A "*" bit in X is selected and assigned a fixed but randomly generated binary constant, $a \in \{0, 1\}$, while the corresponding "*" bit in $I(\hat{t}_i)$ is assigned its complement $\bar{a}$. Meantime, all the "–" bits in $I(\hat{t}_i)$ are replaced by the "*" bits. For example, if X = "1–*" and $a = 0$, then it will be split into X = "1–0" and $I(\hat{t}_i)$ = "1*1". As the number of transitions with "*" bits in the pseudo input variable space enormously outnumbers those in the original input variable space, to simplify the next state and output decoder design, it is lucrative to preserve "*" in the pseudo input variable space whenever free input combinations from the original variable space can be used to produce $I(\hat{t}_i)$. In the search for the next state $\hat{s}_{i+1}$ an input alphabet with j exclusive "*" bits is considered as a cover of 2^{j-1} free input combinations. When two states possess the same highest number of free input combinations, preference will be given to the state that covers the highest number of output combinations in its fan out transitions.

```
Find I(t̂_i)(n, M̂, ŝ_i) {
  if (* is absent in all X ∈ F(ŝ_i)) I(t̂_i) = any X ∈ F(ŝ_i);
  else {
     Select X ∈ I(t_u) with t_u ∈ T(ŝ_i) and ∃X_k = * for 1 ≤ k ≤ n;
     Set I(t̂_i) = X;
     a = random(0,1);
     X_k = a; I(t̂_i)_k = ā ;
     for ( j = 1 to n)
        if ( I(t̂_i)_j = – ) I(t̂_i)_j = *; }
  return I(t̂_i);
}
```

Fig. 4. Finding input alphabet for the watermarked transition.

The above watermarking process is repeated for i = 2 to N until $\hat{t}_N$ is determined. The residual "*" in the input alphabets of all edges will be replaced with "–" and the resultant *STG*(*M*) is the watermarked *STG*($\hat{M}$) and $\hat{\mathbf{X}} = I(\hat{t}_1)I(\hat{t}_2)\cdots I(\hat{t}_N)$. If the overhead of watermarked design is not satisfactory, the entire process can be repeated with an adjusted value of *N*. The overall watermark insertion process is shown in Fig. 5.

For each pseudo input variable added, at least 2^{n-1} potential free input combinations are created in every state transition, where n here refers to the total number of input variables including the pseudo variables. These free input combinations have been consumed in [14] by fixing the value of each pseudo input variable to be "0" consistently for all existing transitions and "1" consistently for the watermarked transition immediately upon its creation. This has not only increased the complexity of

the decoders, but also made the watermarked transition discernible from the pseudo inputs. The introduction of reserved free literal allows the assignments of "*" in the input alphabets of all transitions to be deferred until some input combinations subsumed by it are needed to watermark a transition. The transformation of "–" to "*" in $I(\hat{t}_i)$ when a random assignment is made on "*" serves two important purposes. First, it judiciously preserves the *don't care* inputs in the transitions to optimize the design of next state and output decoders. Second, it allows the same edge to be revisited for watermarking to maximally exploit the free input combinations. This will minimize the required number of pseudo input variables, especially when a long watermark is to be embedded for a strong authorship proof.

FSM_watermarking (M, MSG, m, k, N, Q, K_e) {
  $\boldsymbol{W} = \{w_i\}_{i=1}^{m}$ = **MD**(**encrypt**(MSG, K_e));
  $\boldsymbol{B}$= $\{b_i\}_{i=1}^{m}$ = PNG(K_e, $N \times k$) , $b_i \in [1, N \times k]$;
  $\hat{\mathbf{Y}}$ = **Generate** $\hat{\mathbf{Y}}$ (W, B);
  $\hat{s}_1 \in Q$; $\hat{M} = M$;
  for (i = 1 **to** N) {
    Find $\hat{t}_i$ (Q, i, $\hat{M}$, $\hat{\mathbf{Y}}$, $\hat{s}_i$);
    $\hat{s}_i = D(\hat{t}_i)$; }
  for (each transition $\hat{t} \in \hat{M}$) {
    replace * in $I(\hat{t}$) by –; }
  return $\hat{M}$;
}

Fig. 5. Algorithm for FSM watermarking.

The number of transitions N has no bearing on the probability of coincidence but it has impact on the cost of watermarking. If N is small, the probability of finding compatible outputs from existing transitions is low and more design overhead will be incurred. On the other hand, if N is large, fewer new transitions and pseudo inputs need to be added which will lower the cost of watermarking, but the code sequences required to detect the watermark is long. To avoid introducing an excessive number of unspecified transitions due to the addition of pseudo input variables, N needs to be sufficiently larger than m/k. When $N \approx m$, each output alphabet in $\hat{\mathbf{Y}}$ contains one watermark bit on average and the resultant watermarked design generally possesses acceptably low overhead. As our embedding algorithm can run very quickly even for large FSM, the watermarking process can be repeated for different N to select the least overhead watermarked design with reasonable verification code length. The procedure shown in Fig. 6, is suggested to legitimately limit the number of trials. Let A_{wm_i} denote the area of watermarked FSM with $N = N_i$ at the i-th trial. $N_i = N_{i-1} \pm \delta_i$ and $N_1 \approx m$. N_i that is incremented (or decremented) by δ_i depends on the extent to which $A_{wm_{i-1}}$ is increased (or reduced) over the previous trial. The standard deviation, σ_i, of A_{wm} is defined as:

$$\sigma_i = \sqrt{\frac{1}{i}\sum_{j=1}^{i}(A_{wm_j} - \overline{A_{wm}})^2} \qquad (1)$$

where $\overline{A_{wm}} = \frac{1}{i}\sum_{j=1}^{i} A_{wm_j}$ is the mean area of trial watermarked FSMs. The trial terminates when $\sigma_i / A \le \varepsilon$ or when $N \ge N_{max}$, where ε is a small preset value, A is the area of FSM before watermarking and N_{max} is some preset limit on the verification code length. The least overhead watermarked design from among the trials is selected.

N_adaptation (A, ε, N_{max}, m) {
  i = 1; $N_i \approx m$; $A_{wm_0} = A$; $N = N_i$;
  repeat {
    $\hat{M}$ = **FSM_watermarking** (M, MSG, m, k, N_i, Q, K_e);
    Synthesize $\hat{M}$ and obtain A_{wm_i} ;
    if ($A_{wm_i} > A_{wm_{i-1}}$)
      $N_{i+1} = N_i + \delta_i$;
    else
      $N_{i+1} = N_i - \delta_i$;
    Compute $\overline{A_{wm}}$ and σ_i ;
    $N = N_i$; $i = i + 1$;
  } **until** $\sigma_i / A \le \varepsilon$ **or** $N \ge N_{max}$;
  return $\hat{M}$ with minimum area;
}

Fig. 6. Minimization of FSM watermarking overhead by adaptation of N.

D. *Watermark Detection*

To verify the authorship, one needs to run the watermarked FSM with the input sequence, $\hat{\mathbf{X}} = \{\hat{X}_1, \hat{X}_2, \cdots, \hat{X}_N\}$, applied on state $\hat{s}_1$. If the operation halts before N transitions, the watermark cannot be detected. Otherwise, an output sequence $\tilde{\mathbf{Y}}$ of $N \times k$ bits is obtained. The bits indexed by the set B of m random numbers are selected from $\tilde{\mathbf{Y}}$ to form an ordered sequence $\tilde{W}$. The authorship is proved if $\tilde{W}$ perfectly matches or is highly correlated with the watermark W of the IP owner.

Although the ownership can be authenticated directly by running the watermarked FSM with $\hat{\mathbf{X}}$, it does not permit the IP authorship to be field authenticated by the IP buyers after the watermarked FSM has been implemented into an integrated circuit and packaged. Since only the test signals can be traced after the chip is packaged, the authorship of the watermarked FSM can be verified off chip by making it a part of the test kernel. A sequence of test vectors can be applied serially through the scan-in, S_{in} pin to bring $\hat{M}$ to the designated state $\hat{s}_1$ in the test mode, followed by N designated test vectors that incorporate $\hat{\mathbf{X}}$. The output responses $\tilde{\mathbf{Y}}$ can then be collected serially from a scan-out S_{out} pin externally to verify the authenticity of $\hat{M}$. This convenient way of watermark verification can be performed by the end users provided that scan design is also incorporated in the watermarked IP chip.

Since the scan chain is used as a medium to aid authorship verification of the IP encapsulated in the test kernel, it can also be independently protected by [16], [17] to boost the confidence in positive watermark identification. By watermarking the scan chain of watermarked FSM using the techniques proposed in [16], [17], the aggressor needs additional effort to also

successfully tamper or redesign the test structure to provide the fault coverage of the pirated IP. Failure to detect the scan chain signature alerts malicious tampering or removal of the test structure in attempt to misappropriate the protected IP.

E. *An Illustrative Example*

The STG of a simple FSM to be watermarked is shown in Fig. 7(a). It has five states, represented mnemonically as $Q = \{s_1, s_2, s_3, s_4, s_5\}$. Assume that the encrypted watermark W = "110110". The number of output labels to be mapped, N should be greater than 6/2 = 3 as $m = 6$ and $k = 2$. Let $N = 7$. Suppose the set of six random numbers between 1 and 14 ($k \times N$) generated by the PNG with the IP owner's secret key is $B = \{9, 4, 2, 7, 12, 3\}$.

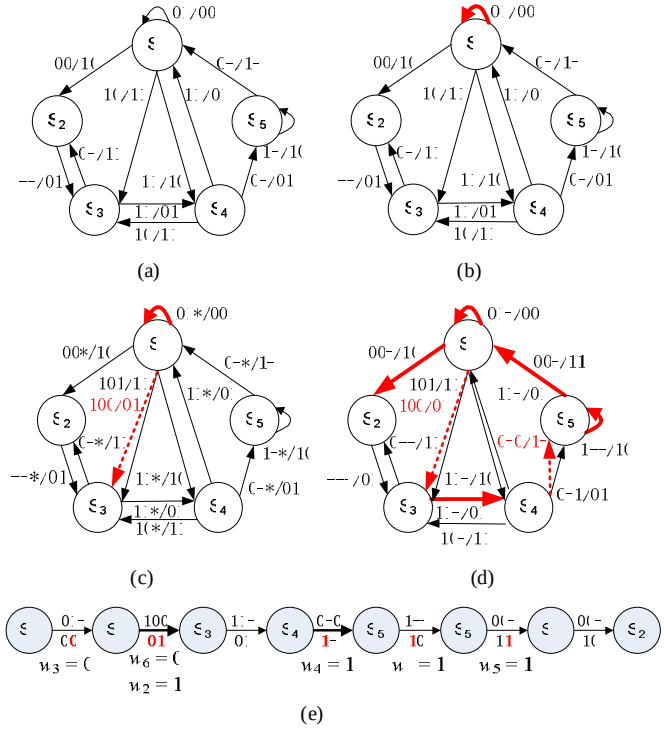

Fig. 7. Example of watermarking on FSM: (a) original FSM, (b) use of existing transition, (c) introduction of pseudo input variable and new transition, (d) the watermarked FSM, (e) excitation of watermarked transitions.

Following the algorithm in Fig. 2, since $2(1-1)+1 = 1 \notin B$, $\hat{y}_{1,1}$ = "–"; since $2(1-1)+2 = 2 = b_3$, $\hat{y}_{1,2} = w_3$ = "0"; $3 = b_6 \Rightarrow \hat{y}_{2,1} = w_6$ = "0"; $4 = b_2 \Rightarrow \hat{y}_{2,2} = w_2$ = "1"; $5 \notin B \Rightarrow \hat{y}_{3,1}$ = "–"; $6 \notin B \Rightarrow \hat{y}_{3,2}$ = "–"; $7 = b_4 \Rightarrow \hat{y}_{4,1} = w_4$ = "1"; $8 \notin B \Rightarrow \hat{y}_{4,2}$ = "–"; $9 = b_1 \Rightarrow \hat{y}_{5,1} = w_1$ = "1"; $10 \notin B \Rightarrow \hat{y}_{5,2}$ = "–"; $11 \notin B \Rightarrow \hat{y}_{6,1}$ = "–"; $12 = b_5 \Rightarrow \hat{y}_{6,2} = w_5$ = "1"; $13 \notin B \Rightarrow \hat{y}_{7,1}$ = "–" and $14 \notin B \Rightarrow \hat{y}_{7,2}$ = "–". Hence, $\hat{\mathbf{Y}}$ = "–0 01 — 1– 1– –1 —".

An arbitrary starting state, $\hat{s}_1 = s_1$, is selected to commence the watermarking process. For $\hat{Y}_1$ = "–0", $C(\hat{s}_1) = \{s_1, s_2, s_4\}$ and none of them has any free input combination. $\hat{s}_2$ can be set to any state of $C(\hat{s}_1)$, says s_1, and $\hat{Y}_1$ = "00". $\hat{t}_1$ is marked by a heavy edge in Fig. 7(b). For $\hat{Y}_2$ = "01", there is no compatible output from $T(\hat{s}_i)$ and $C(\hat{s}_1) = \varnothing$. Since s_3 has the most free input combinations among $D(\hat{s}_2)$, a new transition from s_1 to s_3 is added. As $F(\hat{s}_2) = \varnothing$, a pseudo input variable, x_3, is introduced. It assumes a value of "*" on the inputs of all existing transitions. Suppose the input of transition, $t = (s_1, s_3,$ "10*", "11"), is split into $I(t)$ = "101" and $I(\hat{t}_2)$ = "100". The new transition, $\hat{t}_2 = (s_1, s_3, 100, 01)$, is added into the $STG(M)$ as indicated by a dotted edge in Fig. 7(c). For $\hat{Y}_3$ = "—", $C(\hat{s}_3) = \{s_2, s_4\}$. Both states have equal number of free input combinations but s_4 is preferred over s_2 as s_4 covers more output combinations ("01" and "11") in its fanout transitions than that ("01") of s_2. Therefore, $\hat{t}_3 = (s_3, s_4,$ "11*", "10"). The process continues until all seven transitions are identified. Then all residual "*" in the final STG are changed to "–". The watermarked STG is shown in Fig. 7(d), where the transitions of Cases 1 and 2 are marked by heavy edges and the added transitions of Cases 3 and 4 are marked by dotted edges. Fig. 7(e) shows the complete watermarked sequence of inputs and outputs and the transitional states. The overhead of the synthesized watermarked FSM can be checked at this point. N is modified and the watermarking process is repeated according to Fig. 6 until the terminal criterion is met.

To verify the existence of watermark W, an input sequence, $\hat{\mathbf{X}}$ = ("01–", "100", "11–", "0–0", "1––", "00–", "00–"), "–" $\in \{0,1\}$, is applied on the state s_1. A binary stream $\tilde{W}$ is retrieved from the bit positions, 9, 4, 2, 7, 12, 3 of the output sequence $\hat{\mathbf{Y}}$. If $\tilde{W} = W$ = "110110", the authorship is proved.

IV. Watermark Resilience Analysis

A. *Authorship Credibility*

The credibility of the authorship proof can be evaluated by the probability that an unintended watermark is detected in a design [13]. Suppose that an arbitrary input sequence exits to excite N' ($N' \geq N$) consecutive transitions through the reachable states of an FSM with k output variables. The output sequence of length N' (each output alphabet has k binary bits) will be one of $2^{k \times N'}$ possible solutions. The odds that the output sequence contains the identical watermark bits at the positions specified by the author's signature are:

$$P_c = \frac{2^{k \times N' - m}}{2^{k \times N'}} = \frac{1}{2^m} \tag{2}$$

A longer watermark has a lower probability of coincidence. As m increases, more new transitions may have to be added. The beauty of our method is the input sequence length, N can increase to mitigate the overhead increment without compromising the authorship credibility.

The false positive rate, which is the probability that the watermark is detected in the output sequence under a different random input sequence, can be estimated statistically. If there are $N_C(\tau)$ output sequences detected with at least τ fraction of matched watermark bits when N_T random input sequences are applied, then the false positive rate is determined as:

$$P_\lambda(\tau) = \frac{N_C(\tau)}{N_T} \tag{3}$$

where $0 \leq \tau \leq 1$. To constitute a false positive, $\tau = 1$ since all bits extracted from the specific positions by the detector need to be matched exactly with the watermark bits. As τ reduces, P_λ increases and a threshold of discrimination can be determined

empirically that with certain degree of confidence, the authenticity of the design can be assured by detecting only a fraction of the watermark bits. A suitable error correction scheme can also be considered based on P_λ to correct the partially corrupted output subsequence due to tampering.

P_c and P_λ are important to repudiate the denial of authorship. To show that the output sequences excited by the verification input cannot be obtained by trial-and-error to match the watermark, the claimant needs only to demonstrate that the watermark and the watermarked positions in the output sequence are uniquely generated with a cryptographic one-way function using a secret key in his/her possession, provided that P_c is very low and P_λ is low enough for a sufficiently large number of random tests.

B. *Resilience Analysis*

The following conceivable attacks on watermarking of sequential circuit designs are analyzed with Alice as the IP owner and Bob as the attacker, who attempts to tamper an illegally acquired copy of Alice's watermarked IP.

B.1 Combinational Logic Re-synthesis

Bob may use various logic optimization tools [22], [23] to re-synthesize the combinational logic of watermarked FSM. Such combinational logic re-synthesis operation maintains the inputs/outputs behaviors of flip-flops in the design and has no effect on the STG structure. Therefore, the watermark embedded on the STG is robust against attack by combinational logic re-synthesis.

B.2 Circuit Retiming

Bob may apply retiming transformation [22], [24] to move the latches across the combinational logic blocks of Alice's watermarked FSM without changing the design functionality. Retiming can change the STG structure. Such transformation can be divided into three cases for analysis. (1) Splitting one state into two one-step equivalent states. (2) Merging two one-step equivalent states into one state. (3) Switching between two states that are one-step equivalent. Two states s_i and s_j are said to be one-step equivalent if and only if the two states have the same outputs and the same next state under the same input excitation.

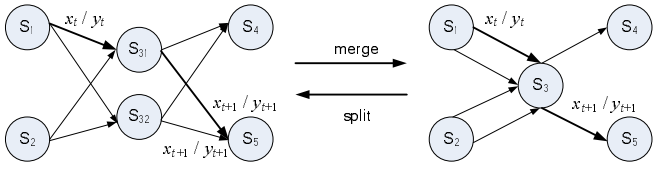

Fig. 8. FSM Retiming.

The consequence of splitting, merging or switching transformation on the outputs retrieved in the watermark detection process can be analyzed by the STG before and after retiming. As an example, let states s_{31} and s_{32} be two generic one-step equivalent states and the transitions (s_1, s_{31}, x_t, y_t) and (s_{31}, s_5, x_{t+1}, y_{t+1}) are traversed in the watermarking process as shown in Fig. 8. Upon retiming, states s_{31} and s_{32} are merged into state s_3. When the sequence $\hat{\mathbf{X}}$ is applied onto the retimed FSM, transitions (s_1, s_3, x_t, y_t) and (s_3, s_5, x_{t+1}, y_{t+1}) are traversed, the same outputs as Alice's watermarked FSM are generated from these two steps. Similarly, splitting or switching operations on the watermarked FSM will not prevent the detection of Alice's watermark. Alice's watermark will not be removed as a state can only be substituted by the state with the same behaviors in retiming transformation.

B.3 State Recoding (*or Assignment*)

Bob may recode the states of Alice's watermarked FSM to remove her watermark. State assignment changes the mnemonic representations of states in Q. It has no effect on the functional specification of FSM [25]. As the watermark is embedded in the state transitions rather than the states, Alice's watermark will survive the state recoding attack.

B.4 Combinational and Sequential Redundancy Removal

When a redundant fault is identified in a sequential circuit, the part of logic can be deleted to simulate the effect of fault. Bob can remove the combinational logic that is not necessary for the correct circuit behavior. This attack has similar effect as the combinational resynthesis attack as far as the sequential behavior is concerned. So it will not affect the embedded watermark.

Elimination of sequential redundancy may change δ and λ while maintaining the I/O behaviors. The sequential redundancies can be categorized into sequentially non-excitable (SNE) and non-distinguishable (ND) faults [26]. An SNE fault is a fault that cannot be excited from any reachable state [26]. As an SNE fault does not affect the reachable part of STG, removal of SNE faults maintains the integrity of reachability information. In our watermarking scheme, all states traversed by $\hat{\mathbf{X}}$ are reachable as long as the starting state, $\hat{s}_1$ is selected as a reachable state after the reset state, s_0. This can be easily guaranteed. As all IOs on the edges of these reachable states are not changed, Alice's watermark can still be detected upon the removal of SNE faults.

Although an ND fault does not affect the I/O behavior, it may change the reachable part of STG. An ND fault can be identified by verifying the equivalence between the watermarked circuit and the circuit obtained by forcing one node in the circuit to a constant value [26]. If they are equivalent, then a stuck-at fault at that node is non-detectable and some redundancy can be removed. The FSM watermark may be partially erased if the ND faults are detected around the circuit corresponding to the added transitions in the watermarking process. However, this attack is expensive since it requires for each node a computation of equivalence between two possibly large sequential circuits. This equivalence is obtained by computing the product machine and its set of reachable states. Even with the use of implicit STG traversal techniques, the applicability of this type of sequential redundancy removal is restricted to small circuits. An ND fault can be excited, but none of the excitation vectors can be extended to a test as its effect can never be observed from any primary output.

B.5 State Reduction

Bob may perform a state reduction on Alice's watermarked FSM based on the identification of sets of compatible states

(compatibles) [26], [27]. A set of states is a compatible if and only if for each input sequence, there is a corresponding output sequence which can be produced by each state in the compatible. All outputs in the transitions are preserved in the reduced FSM even if the states have been substituted by their compatibles. As the watermark is embedded in the transitions instead of the states, our FSM watermarking will survive the state reduction operation. However, a watermark embedded on the states of STG, as in the scheme of [12], is vulnerable to the state reduction operation.

B.6 Transition Elimination

Bob may try to eliminate some transitions in $\hat{t}$. In our watermarking scheme, the existing transitions and the added transitions are indistinguishably utilized for watermarking. There are few added transitions and they are randomly interleaved in the watermarked transition sequence, $\hat{t}$. There is no easy means to eliminate these transitions from the circuit netlist without modifying the correct behaviors of FSM. The time and effort required for a successful attack is almost as good as redesigning the IP function from scratch.

B.7 Removal of Circuitries with Pseudo Inputs

The pseudo inputs, if any, are documented as part of the test or primary inputs in the distributed watermarked IP. Due to their random logic assignments, and the high number of don't cares they introduced, they are well camouflaged after the logic optimization process. Even if Bob knows about the addition of some pseudo inputs, removal of the circuitries connected to these pseudo inputs will cause malfunction to the watermarked FSM. The conflicts arise because the unspecified transitions created by the pseudo inputs can have different outputs or destination states under the same input combinations as the existing transitions upon the removal of the pseudo inputs. For example, in Fig. 7(c), when the pseudo input variable is eliminated by the removal of some subcircuits, there will be two transitions from state s_1 with I/O = 10/11 and 10/01, respectively to state s_3. This is obviously an output conflict, hence such attack is not sustainable.

B.8 Ghost Search

Without tampering Alice's design, Bob may claim his ownership of Alice's FSM by specifying some bits in the output sequence generated by his own selected input sequence to make up his watermark. However, it is computationally infeasible for Bob to reverse the PNG to prove that the positions of these extracted bits are cryptographically related to his signature. Alternatively, he can generate a group of integers with his key using a one-way function and then select the bits from these positions to extract his watermark. Again, it will be computationally infeasible for him to show that the watermark is cryptographically associated with a meaningful ownership message. It is also computationally impractical for Bob to enumerate different sequences of input combinations to match his own watermark to the extracted output sequence of Alice's FSM in his chosen bit positions. The number of trials grows exponentially with the size of FSM. Depending on Bob's selected bit positions, there is no guarantee that such an input sequence can be found even after trying all possible input sequences.

B.9 Addition of Watermark

Bob may embed his own watermark into Alice's watermarked design to claim his ownership, if he has the necessary tools and knowledge of the watermarking process. Owing to the resilience of the proposed watermarking scheme against watermark erasure without changing the properties of FSM, even if Bob can succeed in adding his own watermark into Alice's watermarked FSM, Bob's watermarked design will contain Alice's watermark. Therefore, Alice can still correctly retrieve her watermark bits from Bob's watermarked design but the reverse is not possible for Bob.

If the protected IP is distributed at the gate-level, Bob would have to first recover the STG from the netlist, which is computationally impractical for large designs [11]. Additionally, Bob needs to repeat the entire watermarking and optimization process to ensure that the overhead is acceptable. Yet, this problem can be solved by using a secure third party (entity), e.g., a legal firm or a watermarking governing body. In this case, Alice will generate a time-stamped authenticated signature, and keep it at an authorized legal firm. This firm will keep a record of such signatures and the date it was generated, which can be used in front of a court to show the exact time the watermark was generated and embedded in any future dispute. The overall IP watermarking framework is depicted in Fig. 9.

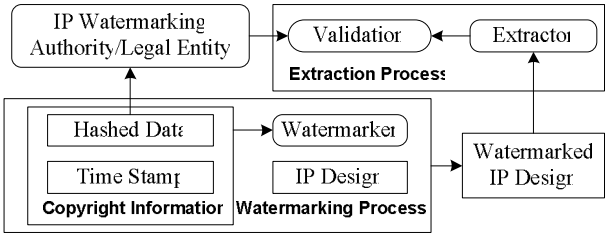

Fig. 9. Watermarking with third party keeping a time-stamped signature.

V. Experimental Results

The experimentation is performed on the circuits, which are described in KISS2 format [23], from the IWLS'93 benchmark set and some FSM designs from the ISCAS'89 benchmark set. The FSM watermarking scheme is implemented using the C++ language. 64-bit and 128-bit watermarks were embedded into each FSM design. Using the SIS [23] tool, state minimization and state assignment are carried out on the original and watermarked designs. The optimized FSM designs are synthesized using the algebraic script from SIS and technology mapped to the Mississippi State University standard cell library. All experiments were run on a 750MHz Sun UltraSPARC-III with Solaris operating system and 2 GB of memory.

Table II summarizes experimental results conducted on ISCAS'89 and IWLS'93 benchmark designs. The columns "$|Q|$", "n" and "k" are the numbers of states, input variables and output variables of each FSM design, respectively. "A" and "D" are the area and delay, respectively,of the optimized design as reported by SIS [23] before watermarking. "P" is the estimated power in μW obtained by using GENERAL delay model [23] with 20MHz clock and 5V supply. Each design is watermarked with the first 64 and 128 bits of SHA-1 hash values of the ownership information. Different lengths of verification code sequence, N

have been experimented with $N_1 = 80$ and $N_{max} = 600$. $\delta_i = 20$ when $N_i < 100$ and $\delta_i = 100$ when $N_i \geq 100$. Typically, σ_i/A converges to $\varepsilon = 0.05$ in less than five trials. The value of "N" indicated is the one that produces the least area overhead watermarked FSM design. For most designs tested, only one pseudo input variable is introduced in the watermarking process while no pseudo input variable is needed for the designs, "ex4", "ex1" and "sand". "n_a" denotes the number of new transitions added onto the watermarked STG. ΔA, ΔD and ΔP are the percentage area, delay and power overheads, respectively. A negative percentage implies that watermarking has actually improved the performance. In general, more new transitions have been added onto the designs with 128-bit watermark than with 64-bit watermark. The performance overheads decrease as the size of FSM increases. For the six larger designs (twelve watermarked designs), the average area has increased by 4.23% but the average timing and power have actually improved by 0.52% and 0.33%, respectively. It is conjecture that the watermarking overheads will become negligible for FSMs with many more states and input and output variables than those simulated.

TABLE II STATISTICS FOR ISCAS'89 AND IWLS'93 BENCHMARKS

circuit	$\|Q\|$	n	k	A	D	P	m	N	n_a	ΔA	ΔD	ΔP
s27	6	4	1	824	7	307	64	600	2	3.9	5.7	-6.2
							128	300	2	3.9	5.7	-6.2
s208	18	11	2	1912	13	672	64	100	4	3.8	-9.2	6.0
							128	100	4	4.2	-9.2	10.3
s386	13	7	7	2512	13	842	64	100	5	20.7	7.7	9.7
							128	600	5	22.6	10.8	15.2
s832	25	18	19	5144	19.8	1714	64	40	4	10.0	2.0	6.5
							128	200	6	18.2	-4.0	16.2
s510	47	19	7	5528	18.4	2021	64	100	4	6.5	-3.3	2.8
							128	200	7	13.5	-3.3	10.1
s820	25	18	19	6112	20.6	2089	64	200	3	-10.0	-4.9	-17.3
							128	300	4	3.8	-2.9	-3.0
s1488	48	8	19	105	23	3391	64	200	2	3.0	5.2	10.8
							128	300	4	5.9	1.7	12.8
s1494	48	8	19	109	24.8	3746	64	300	4	-0.4	-2.4	-6.5
							128	300	5	8.3	-1.6	4.4
bbara	10	4	2	1112	10.4	433	64	600	3	8.6	-11.5	-4.7
							128	600	5	17.3	-1.9	1.4
dk15	8	3	5	1440	10	556	64	500	1	0.6	4.0	4.3
							128	500	1	8.9	12.0	2.7
ex4	14	6	9	1584	10.8	553	64	300	11	30.8	3.7	36.7
							128	600	9	22.7	1.9	26.9
opus	10	5	6	1768	10.8	612	64	400	4	10.0	11.0	4.9
							128	400	9	21.7	18.5	16.5
sse	16	7	7	2560	12.2	905	64	400	3	16.3	11.5	15.1
							128	500	6	13.1	16.4	-2.5
ex1	20	9	19	4360	16	1439	64	300	5	18.7	16.3	15.0
							128	200	10	32.5	20.0	22.2
s1	20	8	6	5112	16.6	1844	64	200	4	-7.5	1.2	-8.0
							128	500	7	9.4	21.7	4.7
tbk	32	6	3	5352	19.6	1870	64	400	2	-2.4	3.1	-13.9
							128	200	6	0.0	-0.01	-14.7
styr	30	9	10	8408	22.2	3028	64	60	5	7.4	0.0	-4.2
							128	200	5	3.0	-11.7	-1.2
sand	32	11	9	9096	24.8	3128	64	400	4	1.0	-13.7	-1.4
							128	400	5	3.3	0.0	-6.5
planet	48	7	19	9784	21	3454	64	100	5	1.4	0.95	-3.5
							128	100	6	12.1	-2.9	7.9
ram_test	72	16	24	9840	23.8	3563	64	600	3	-7.0	-16.7	-16.0
							128	600	4	-7.0	-10.1	-13.2
scf	121	27	56	12640	23.8	3705	64	400	13	11.8	9.2	0.8
							128	400	15	17.3	10.1	6.5

According to (2), the probabilities of coincidence, $P_c = 5.42 \times 10^{-20}$ and 2.49×10^{-39} for $m = 64$ and 128, respectively. The false positive rate P_λ is determined empirically by applying 1000 randomly generated input code sequences of length N onto each watermarked FSM at the watermarked starting state. None of the output sequence was detected with a perfectly matched watermark for each watermarked FSM, i.e., $P_\lambda(\tau = 1) = 0$ for all watermarked designs. It is thought to be reasonable that a sufficiently low probability is adequate to prove the authorship and make the denial attacks unsustainable. Hence, we reduce the watermark correlation from 100% to 75% match. It was found that for $\tau = 0.75$, $P_\lambda = 0$ for all the watermarked designs. When τ is reduced to 0.7, only a small number of watermarked designs has $P_\lambda > 0$. Based on these results, it is reasonable to assume that when more than three quarters of watermark bits are matched, the authorship proof is still veracious.

We used the SIS tool and the same technology library to synthesize the designs watermarked by the method in [14] and compared their areas and delays with those of our proposed (abbreviated as Prop.) FSM watermarking method in Table III. ΔA, ΔD and ΔP are the percentage reductions of area, delay and power, respectively, of our proposed scheme over those of [14]. It is evident that most designs watermarked by our method have lower area, timing and power overheads.

TABLE III COMPARISON WITH FSM WATERMARKING METHOD IN [14]

circuit	m	Area			Delay			Power		
		[14]	Prop.	ΔA	[14]	Prop.	ΔD	[14]	Prop.	ΔP
s27	64	1064	856	19.6	7.8	7.4	5.1	411	288	30
	128	1064	856	19.6	7.8	7.4	5.1	411	288	30
s208	64	3680	1984	46.1	15.8	11.8	25.3	1314	712	45.8
	128	5248	1992	62.0	19.8	11.8	40.4	1968	741	62.3
s386	64	3976	3032	23.7	17.8	14.0	12.5	1257	923.	26.6
	128	4592	3080	32.9	16.2	14.4	11.1	1479	970	34.4
s832	64	6256	5656	9.6	21.0	20.2	3.8	1939	1826	5.83
	128	6904	6080	11.9	21.6	19.0	12.0	2026	1991	1.73
s510	64	7272	5888	19.0	21.2	17.8	16.0	2641	2077	21.4
	128	7816	6272	19.8	19.4	17.8	8.2	2805	2226	20.6
s820	64	6208	5504	11.3	20.6	19.6	4.9	1944	1727	11.2
	128	7352	6344	47.0	21.4	20.0	6.5	2183	2025	7.24
s1488	64	12032	10864	9.7	23.2	24.2	-4.3	3877	3758	3.07
	128	14120	11168	20.9	25.6	23.4	8.6	4399	3825	13.0
s1494	64	12488	10856	13.1	29.0	24.2	16.6	4166	3503	15.9
	128	12816	11808	7.9	25.2	24.4	3.2	4143	3912	5.58
bbara	64	2512	1208	51.9	13.2	9.2	30.3	888	412	53.6
	128	2512	1304	48.1	13.2	10.2	22.7	888	439	50.6
dk15	64	2104	1448	31.2	11.6	10.4	10.3	738	581	21.3
	128	2504	1568	37.4	13.6	11.2	17.6	946	571	39.6
ex4	64	2104	2072	1.5	11.4	11.2	1.8	756	756	0.0
	128	3008	1944	35.4	13.0	11.0	15.4	1077	702	34.8
sse	64	4216	2976	29.4	15.4	13.6	11.7	1474	1041	29.4
	128	4496	2896	35.6	15.2	14.2	6.6	1460	882	39.6
ex1	64	5632	5176	8.1	16.0	18.6	-16.3	1764	1654	6.24
	128	6056	5776	4.6	18.4	19.2	-4.3	1880	1758	6.49
s1	64	5760	4728	17.9	16.8	16.8	0.0	2109	1695	19.6
	128	7376	5592	24.2	19.6	20.2	-3.1	2675	1935	27.7
sand	64	8944	9184	-2.68	20.4	21.4	4.9	3013	3086	2.37
	128	10952	9392	14.2	21.6	24.8	-14.8	3674	2925	20.4
planet	64	11552	9920	16.5	22.6	21.2	6.2	3787	3332	12.0
	128	11712	10968	6.35	25.2	20.4	19.0	4064	3725	8.34
styr	64	9240	9032	2.25	25.0	22.2	11.2	3014	2900	3.78
	128	10216	8656	15.3	24.2	19.6	19.0	3407	2992	12.2
scf	64	13568	14128	-4.1	24.4	26	-6.6	3679	3733	-1.5
	128	13880	14824	-6.8	22.8	26.2	-14.9	3625	3945	-8.8

We also compared our watermarking method with Oliveira's [12] FSM watermarking scheme in Table IV. As the same synthesis tool and technology library were used, the area and delay results are excerpted from [12] for those circuits provided with both BLIF and KISS2 formats in the benchmark suite and have comparable literal counts in their original designs before watermarking. In Table IV, the area is measured in terms of the number of literals to be consistent with [12]. All designs watermarked by our method have consistently lower area and timing overheads than [12]. It should also be noted that the watermarking method of [12] does not survive the state reduction operation (cf. III.B.3).

TABLE IV COMPARISON WITH FSM WATERMARKING METHOD IN [12]

Circuit	*M*	Area			Delay		
		[12]	proposed	ΔA (%)	[12]	proposed	ΔD (%)
s27	64	297	51	82.8	21.8	7.4	66.1
	128	541	51	90.6	25.4	7.4	70.9
s208	64	308	164	46.8	15.6	11.8	24.4
	128	441	170	61.5	19.8	11.8	40.4
s386	64	444	248	44.1	17.8	14.0	21.3
	128	644	258	59.9	23.0	14.4	37.4
s499	64	879	247	71.9	34.6	13.4	61.3
	128	1230	308	75.0	39.6	16.4	58.6
s832	64	680	486	28.5	22.2	20.2	9.0
	128	804	518	35.6	24.4	19.0	22.1
s510	64	581	512	11.9	20.6	17.8	13.6
	128	688	545	20.8	21.0	17.8	15.2
s820	64	669	463	30.8	26.4	19.6	25.8
	128	814	539	33.8	24.0	20.0	16.7
s1488	64	1318	968	26.6	33.4	24.2	27.5
	128	1495	981	34.4	31.0	23.4	24.5
s1494	64	1329	945	28.9	32.8	24.2	26.2
	128	1547	1050	32.1	31.4	24.4	22.3

In Table V, we also compare our FSM watermarking method with the FSM watermarking method [13]. For consistency, the designs are watermarked with the same length of watermark as [13] and synthesized using the same MSU script [23] from SIS. The watermark length "*m*" used by [13] is design dependent and can be determined by the product of the number of output variables $|\Delta|$ of the watermarked design and the minimum number of watermarked transitions $n_{\min}$ needed to satisfy the required probability of coincidence (P_u in [13]). It is evident that our method incurs lower area overhead than [13] for the same constraint on the watermark robustness. Note here that no delay statistics are provided in [13] to compare with.

TABLE V COMPARISON WITH FSM WATERMARKING METHOD IN [13]

Circuit	*m*	Area		
		[13]	proposed	ΔA (%)
s27	36	1.53k	0.62k	59.5
bbara	30	2.01k	1.05k	47.8
dk14	35	1.84k	1.74k	5.4
styr	40	10.69k	7.54k	29.5
bbsse	70	2.62k	2.46k	6.1
cse	35	4.08k	3.62k	11.3
sse	21	2.43k	2.34k	3.7
ex1	76	5.55k	4.38k	21.1
ex1	38	5.40k	4.06k	24.8
scf	112	21.02k	14.4k	31.5

As SIS tool can only read STG in KISS2 format, to show the applicability of our method on large designs, we use GenFSM [28] to generate ten arbitrary STGs of hundreds to thousands of transitions for experimentation by specifying the number of inputs/outputs and states. These FSMs can all be watermarked by the proposed method within one second. The synthesis results are shown in Table VI, where the column "*T*" is the number of transitions of the generated FSM. The largest design has an area of 47721 literals which is much larger than the largest design in [12], which has only 19258 literals. On average, for the128-bit watermark, the area increases by 0.16%, and the delay and power decreases by 2.1% and 0.4%, respectively.

TABLE VI STATISTICS OF WATERMARKING ON FSMS GENERATED BY GENFSM

FSM	*n*	*k*	|*Q*|	*T*	A	D	P	*m*	ΔA	ΔD	ΔP
F1	5	10	10	320	2415	35.2	4456	64	-0.0	17.0	-3.8
								128	-0.6	5.1	-5.6
F2	5	10	14	448	3063	37	6022	64	2.3	3.2	0.18
								128	0.8	2.16	-0.25
F3	3	8	80	640	3990	68.2	5217	64	1.25	-5.57	-1.30
								128	0.10	-17.6	-3.05
F4	3	7	100	800	4552	69.6	5983	64	-0.62	-8.05	8.14
								128	1.10	-8.3	7.24
F5	3	7	200	1600	8875	43	11054	64	-1.18	-2.79	-1.38
								128	0.7	-5.12	-0.62
F6	3	6	300	2400	11625	40.2	13826	64	0.03	-0.5	-2.05
								128	-0.66	0.5	-1.5
F7	3	6	350	2800	13249	45.4	15516	64	0.79	-0.44	-1.51
								128	0.34	1.32	-1.64
F8	3	5	400	3200	14275	44	16290	64	-0.32	-2.3	0.14
								128	-0.29	-2.3	0.84
F9	4	5	500	8000	38388	100.8	39845	64	0.12	1.98	-0.31
								128	-0.02	2.18	0.11
F10	4	5	600	9600	47721	102	46751	64	0.31	-4.31	1.07
								128	0.24	1.18	0.11

VI. CONCLUSIONS

This paper presents a new robust dynamic watermarking scheme by embedding the authorship information on the transitions of STG at the behavioral synthesis level. The proposed method offers a high degree of tamper resistance and provides an easy and noninvasive copy detection. The FSM watermark is highly resilient to all conceivable watermark removal attacks. The redundancy in the FSM has been effectively utilized to minimize the embedding overhead. By increasing the length of input code sequence for watermark retrieval and allowing the output compatible transitions to be revisited to embed different watermark bits, the watermarks are more randomly dispersed and better concealed in the existing transitions of FSM. The new approach to the logic state assignments of pseudo input variables also makes it infeasible to attack the watermarked FSM by removing the pseudo inputs. Without compromising the watermark strength, the length of verification code sequence can be adapted to reduce the area overhead of watermarked design to a reasonable bound within a preset number of iterations. Our experimental results show that the watermarking incurs acceptably low performance overheads and possesses very low possibility of coincidence and false positive rate.

Similar to other FSM watermarking schemes [12]-[14], this method is not applicable to some ultra high speed designs that do

not have an FSM. Fortunately, regular sequential functions are omnipresent in industrial designs [13], making FSM watermarking a key research focus for dynamic watermarking. One recommendation to overcome such limitation is to augment it with combinational watermarking scheme [5] applied simultaneously or on different levels of design abstraction to realize hierarchical watermarking [9], [10]. The watermarked FSM can be fortified by a scan chain watermarking [16], [17] to enable the authorship to be easily verified even after the protected IP has been packaged. While the robustness of the authorship proof lies mainly on the watermarked FSM, the auxiliary post-synthesis scan-chain reordering serves as an intruder-alert for the misappropriation of sequential design under test and increases the effort level required to successfully forge a testable IP without being detected. Even if the scan chain is removed or deranged by the aggressor, the more robust FSM watermark remains intact and detectable on chip.

REFERENCES

[1] Intellectual Property Protection Development Working Group, Intellectual Property Protection: Schemes, Alternatives and Discussion. VSI Alliance, White Paper, Version 1.1, August 2001.

[2] I. Hong and M. Potkonjak, "Techniques for intellectual property protection of DSP designs," *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, Seattle, Washington, USA, vol.5, May 1998, pp. 3133-3136.

[3] A. T. Abdel-Hamid, S. Tahar, and E. M. Aboulhamid, "A survey on IP watermarking techniques," *Design Automation for Embedded Systems*, vol. 10, Springer Verlag, July 2005, pp. 1-17.

[4] D. Kirovski, Y. Y. Hwang, M. Potkonjak and J. Cong, "Protecting combinational logic synthesis solutions," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Syst.*, vol. 25, no. 12, Dec. 2006, pp. 2687-2696.

[5] A. Cui, C. H. Chang and S. Tahar, "IP watermarking using incremental technology mapping at logic synthesis level," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Syst.*, vol. 27, no. 9, Sept. 2008, pp. 1565-1570.

[6] A. Cui and C. H. Chang, "Stego-signature at logic synthesis level for digital design IP protection," *Proc. IEEE Int. Symp. on Circuits and Syst*., Kos, Greece, May 2006, pp. 4611-4614.

[7] A. Cui, C. H. Chang, "Watermarking for IP Protection through Template Substitution at Logic Synthesis Level," *Proc. IEEE Int. Symp. on Circuits and Syst.*, New Orleans, USA, May 2007, pp. 3687-3690.

[8] A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang and G. Wolfe, "Constraint-based watermarking techniques for design IP protection," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Syst.*, vol. 20, no. 10, Oct. 2001, pp. 1236-1252.

[9] H. J. Kim, W. H. Mangione-Smith, and M. Potkonjak, "Protecting ownership rights of a lossless image coder through hierarchical watermarking," *Workshop on Signal Processing Systems*, Cambridge, Massachusetts, USA, Oct. 1998, pp. 73-82.

[10] A. Rashid, J. Asher, W. H. Mangione-Smith, and M. Potkonjak, "Hierarchical watermarking for protection of DSP filter cores," in *Proceedings IEEE Custom Integrated Circuits Conference*, New York, USA, May 1999, pp. 39-42.

[11] A. L. Oliveira, "Robust techniques for watermarking sequential circuit designs," *Proc. IEEE/ACM Design Automation Conf.*, New Orleans, Louisiana, USA, June 1999, pp. 837-842.

[12] A. L. Oliveira, "Techniques for the creation of digital watermarks in sequential circuit designs," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Syst.*, vol. 20, no. 9, Sept. 2001, pp. 1101-1117.

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

[14] A. T. Abdel-Hamid, S. Tahar and E. M. Aboulhamid, "A public-key watermarking technique for IP designs," *Proc. Design, Automation and Test in Europe*, vol. 1, Munich, Germany, Mar. 2005, pp. 330-335.

[15] A. Cui and C. H. Chang, "Intellectual property authentication by watermarking scan chain in design-for-testability flow," in *Proc. IEEE Int. Symp. on Circuits and Syst*., Seattle, USA, May 2008, pp. 2645-2648.

[16] A. Cui and C. H. Chang, "An improved publicly detectable watermarking scheme based on scan chain ordering," in *Proc. IEEE Int. Symp. on Circuits and Syst.*, Taipei, Taiwan, May 2009, pp. 29-32.

[17] C. H. Chang and A. Cui, "Synthesis-for-Testability Watermarking for Field Authentication of VLSI Intellectual Property," *IEEE Trans. on Circuits and Systems-I*, vol. 57, no. 7, July 2010, pp. 1618-1630.

[18] J.-K. Rho, G. D. Hachtel, F. Somenzi and R. M. Jacoby, "Exact and heuristic algorithms for the minimization of incompletely specified state machines," *IEEE Trans.on Computer-Aided Design of Integrated Circuits and Syst.*, vol. 13, no. 2, Feb. 1994, pp. 167–177.

[19] J. M. Pena and A. L. Oliveira, "A new algorithm for exact reduction of incompletely specified finite state machines," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Syst.*, vol. 18, no. 11, Nov. 1999, pp. 1619 – 1632.

[20] T. Kam, T. Villa, R. Brayton and A. Sangiovanni-Vincentelli, "A fully implicit algorithm for exact state minimization," *Proc. ACM/IEEE Design Automation Conference*, Piscataway, New Jersey, USA, June, 1994, pp. 684–690.

[21] A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1996.

[22] G. De Micheli, *Synthesis and Optimization of Digital Circuits*. McGraw-Hill, 1994.

[23] E. M. Sentovich, K. J. Singh, C. Moon, H. Savoj, R. K. Brayton and A, Sangiovanni-Vincentelli, "Sequential circuit design using synthesis and optimization," *Proc. IEEE International Conference on Computer Design: VLSI in Computers and Processors*, Cambridge, MA, USA, Oct. 1992, pp. 328 – 333.

[24] R. K. Ranjan, V. Singhal, F. Somenzi and R. K. Brayton, "On the optimization power of retiming and resynthesis transformations," *Proc. IEEE/ACM International Conference on Computer-Aided Design*, Nov 1998, San Jose, California, USA, pp. 402 – 407.

[25] D. Chen, M. Sarrafzadeh and G. K. H. Yeap, "State encoding of finite state machines for low power design," *Proc. IEEE Int. Symp. on Circuits and Syst.*, Seattle, Washington, USA, April 1995, pp. 2309 - 2312.

[26] T. Kam, T. Villa, R. Brayton and A. Sangiovanni-Vincentelli, *Synthesis of FSMs: Functional Optimization*. Kluwer Academic Publishers, The Netherlands, 1997.

[27] R. J. Kyung, G. D. Hachtel, F. Somenzi and R. M. Jacoby, "Exact and heuristic algorithms for the minimization of incompletely specified state machines," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Syst.*, vol. 13, no. 2, Feb. 1994, pp. 167 – 177.

[28] C. Pruteanu, C. Haba, "GenFSM, a Finite State Machine Generation Tool", in *Proc. 9th International Conference on Development and Application Systems*, Suceava, Romania, May 22-24, 2008, pp. 165-168.

Aijiao Cui (S'06-M'10) received her B.Eng. degree and M.Eng. degree in Electronics from Beijing Normal University, Beijing, China, in 2000 and 2003, respectively, and her PhD degree in Electrical and Electronic Engineering from Nanyang Technological University, Singapore in 2009. From July 2003 to December 2004, she was a lecturer in Beijing Jiaotong University. She worked as a Research Fellow in Peking University ShenZhen SOC Lab, China from 2009 to 2010 before her current appointment as Assistant Professor in Harbin Institute of Technology Shenzhen Graduate School since August 2010. Her current research interests include digital watermarking techniques for IP protection and Design-for-Testability techniques.

Chip-Hong Chang (S'92–M'98–SM'03) received his B.Eng. (Hons) from National University of Singapore in 1989, and his M.Eng. and Ph.D. from Nanyang Technological University (NTU), Singapore in 1993 and 1998, respectively. He served as Technical Consultant in industry prior to joining the School of Electrical and Electronic Engineering, NTU in 1999, where he is now an Associate Professor. He holds joint appointments at the university as Assistant Chair of Alumni, School of EEE since June 2008, Deputy Director of the Centre for High Performance Embedded Systems (CHiPES) since 2000 and Program Director of the Centre for Integrated Circuits and Systems (CICS) from 2003-2009. His current research interests include low power arithmetic circuits, digital filter design, application specific digital signal processing, and digital watermarking for IP protection. He has published three book chapters and more than 150 research papers in refereed international journals and conferences. Dr. Chang serves as the the Associate Editor of the IEEE Transactions on Circuits and Systems-I since 2010, Editorial Advisory Board Member of the Open Electrical and Electronic Engineering Journal since 2007, the Editorial Board Member of Journal of Electrical and Computer Engineering since 2008, and the Guest Editor for the special issue of the Journal of Circuits, Systems and Computers in 2010. He also served in several international conference advisory and technical program committee. His name has been listed in several international biographical records, including the Marquis Who is Who in the World, Who's who in Science and Engineering, Dictionary of International Biography, and the Charter Fellow of Advisory Directorate International of the American Biographical Institute, Inc. He is a Fellow of the IET.

Sofiène Tahar (M'96-SM'07) received the Diploma degree in computer engineering from the University of Darmstadt, Germany, in 1990, and the Ph.D. degree with distinction in computer science from the University of Karlsruhe, Germany, in 1994. Currently, he is a Professor and Research Chair in Formal Verification of System-on-Chip at the Department of Electrical and Computer Engineering, Concordia University, Montreal, QC, Canada. He has made contributions and published papers in the areas of formal hardware verification, system-on-chip verification, analog and mixed signal circuits verification, and probabilistic, statistical and reliability analysis of systems. Dr. Tahar is the founder and director of the Hardware Verification Group at Concordia University. In 2007, he was named University Research Fellow upon receiving Concordia University's Senior Research Award. Dr. Tahar is a Professional Engineer in the Province of Quebec. He has been organizing and involved in program committees of various international conferences in the areas of formal methods and design automation.

Amr T. Abdel-Hamid (S'95-M'05) received his bachelor of Communications and Electronics Engineering from the Faculty of Engineering, Cairo University, Egypt, in 1997, his MaSc. and Ph.D. degrees from Faculty of Electrical and Computer Engineering, Concordia University, Montreal, Canada in 2001, and 2006 respectively. Currently, he is an Assistant Professor at the German University in Cairo, Egypt, since 2006. He has made contributions and published papers in the areas of formal hardware verification, system-on-chip verification, IP watermarking, Protocol Verification, Mobile and Social Network Applications.