INTELLECTUAL PROPERTY PROTECTION OF SEQUENTIAL CIRCUIT DESIGN USING FSM WATERMARKING

S.Chaitanya
Assistant Professor, Dept. of ECE,
Christu Jyothi Institute of Technology & Science
Jangoan, Telangana, India
Email: schaitanya15@gmail.com

Abstract— A sequential circuit design is done by using Finite State Machines (FSM's). This paper proposes a new FSM watermarking scheme by which the authorship information can be made into a non-redundant property of FSM. In the existing free transitions of state transition graph (STG), the watermark bits are interwoven into the outputs to overcome the vulnerability to state removal attack and to minimize the design overhead. Other than conventional transition based STG watermarking, pseudo input variables have been reduced and made functionally indiscernible by the notion of reserved free literal. To minimize the overhead of watermarking and make the watermarked FSM fallible upon removal of any pseudo input variable the assignment of reserved literals is exploited. A convenient and direct 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'3 benchmark sets show acceptably low overheads with higher tamper resilience and stronger authorship proof in comparison with related watermarking schemes for sequential functions

Keywords— Finite state machine (FSM), intellectual property (IP) protection, IP watermarking, sequential design, state transition graph (STG).

I. INTRODUCTION

As reuse based design methodology has taken hold, the very large scale integration (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 [9], [10] or at the design-for-testability 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 a FSM design is usually specified by a 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 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 the existing FSM watermarking scheme to be overcome in this paper are discussed in the next section.

The proposed watermarking scheme thus makes the authorship proof harder to erase and the IP authorship easier to verify. The rest of this 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 this paper.

II. Related Work

The notion of constraint-based watermarking, first proposed by Hong and Potkonjak [2], has now been widely applied to embed authorship signatures 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 a scheme therefore has high resilience against tampering at lower abstraction levels. A 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.

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 binary decision diagram-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. A 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 follows, a more robust technique of transition-based FSM watermarking is proposed to overcome the shortcomings of the above methods. Provisions are 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 a FSM is given in [19] as follows.

Definition 1: A FSM is a tuple $M = (\sum, \Delta, Q, s_0, \delta, \lambda)$, where $\sum$ 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.

$\delta(s, X)$: $Q \times \sum \rightarrow Q \cup \{\emptyset\}$ is the state transition function and $\lambda(s, X)$: $Q \times \sum = \Delta \cup \{\tau\}$ is the output function, where Ø 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 $\exists X \in \sum s.t. s_j = \delta(s_i, X)$. The application of X on s_i also

produces an output, $Y = \lambda(s_i, X) \in \Delta$. For a 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 $\sum$ and Δ, a lower case letter to denote an input or output variable in $\{0, 1, -\}$, and $y_{i,j}$ to address the jth bit of the ith alphabet, Y_i.

FSMs are usually designed with their STG. A 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 a 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 a 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, $\hat{Y} = \{\hat{Y}_1, \hat{Y}_2, \ldots, \hat{Y}_N\}, \hat{Y}_i \in \Delta$, from the watermarked design $\widehat{M}$ by applying a specific input sequence, $\hat{X} = \{\widehat{X_1}, \widehat{X_2}, \ldots, \widehat{X_N}\}$, $\widehat{X_l} \in \Sigma$, on a state, $\hat{S} \in Q$, such that $\hat{Y} = \lambda(\hat{s}, \hat{X}) = \lambda\big(\delta\big(\delta\big(\ldots \delta(\hat{s}, \hat{X}) \ldots\big), \hat{X}_{N-1}\big), \hat{X}_N\big)$. The watermark synthesis process requires that the outputs of $\widehat{M}$ be compatible with the outputs of M for every input symbol,$\hat{X} \in \Sigma$, and output mappings of $\widehat{M}$ for every input symbol, $\widehat{X_l} \in \Sigma \; \forall i = [1, N]$, be dictated by a signature that identifies the ownership of a design. The signature is cryptographically generated with a secret key so that $\hat{Y} = \lambda(\hat{s}, \hat{X})$ becomes a unique property of $\widehat{M}$.

In [13] and [14], the length N of $\hat{X}$ and $\hat{Y}$ is equal to m/k, where m is the watermark length and k is the number of output variables of a 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).

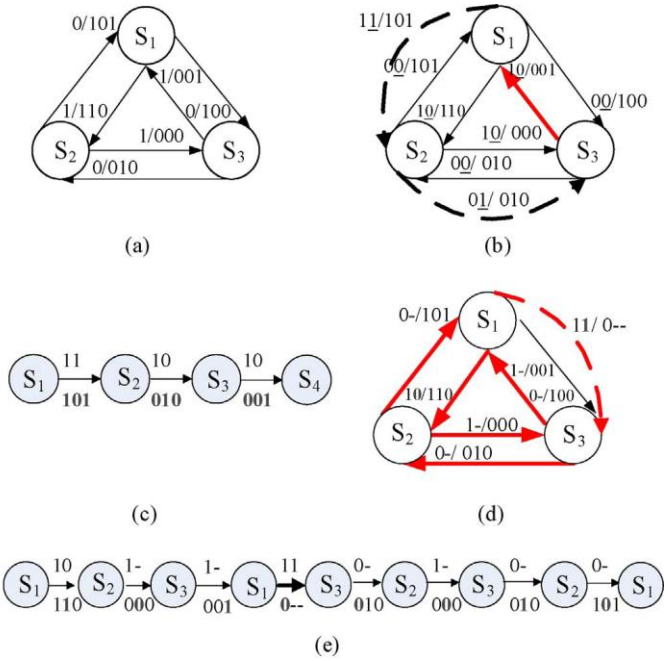

Fig. 1. Watermark embedding on transitions of STG. (a) Original STG. (b) Watermarked STG by the scheme in [14]. (c) Excitation of watermarked transitions of STG in (b). (d) Watermarked STG by proposed scheme. (e) Excitation of watermarked transitions of STG in (d).

A 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), and 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 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 $\hat{X}$ 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 $\hat{Y}$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 eight 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 $\hat{X}$ 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$w_i \epsilon \{0,1\}$.

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.,$b_i \epsilon [1, N \times K] \forall\ i$ = 1, 2, . . . , m and$b_i \neq b_j\ \forall\ i \neq j$. The length N of sequence $\hat{X}$ is determined empirically. The purpose of B is to randomly disperse the m watermark bits into$\hat{Y}$. 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 jth bit of $\hat{Y}_i \in \hat{Y}$. 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.

TABLE I
INTERSECTION OF TWO TERNARY VARIABLES

∩	0	1	-
0	0	Ø	0
1	Ø	1	1
-	0	1	-

```
Generate Ŷ (W, B) {
    Ŷ = {ŷ_{i,j}},  i∈[1, N], j∈[1, k];
    for (i = 1 to N) {
      for (j = 1 to k) {
          ŷ_{i,j} = –;
          for (l = 1 to m) {
              if ((i–1)k + j = b_l) {
                  ŷ_{i,j} = w_l ;
                  break; } }
      } } }
    return Ŷ ;
}
```

Fig. 2. Generation of watermarked output sequence.

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, $\hat{Y} = \hat{Y}_1 \hat{Y}_2 \ldots \hat{Y}_N = \hat{y}_{1,1} \ldots \hat{y}_{1,1} \hat{y}_{2,1} \ldots \hat{y}_{2,k} \ldots \hat{y}_{N,1} \ldots \hat{y}_{N,k}$. The mapping from W to $\hat{Y}$ is injective but not surjective. The value of each bit $\hat{y}_{i,j}$ in $\hat{Y}$ 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 $\hat{}Yi$ and a source state $\hat{}si$, the destination state $\hat{}si$+1 of watermarked transition $\hat{}ti$ 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., $x \cap y \_= \emptyset$. 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 $X \cap Y = \{xi \cap yi\}$ has a null value.
Starting with i = 1, an arbitrary state, $\hat{}s1 \in Q$, is selected.
Let $T(\hat{}si)$ be the set of transitions emanating from a state, $\hat{}si$.
A set of transitions $C(\hat{}si)$ that is output compatible with $\hat{}Yi$ is sought, i.e., $C(\hat{}si) = \{ti \in T(\hat{}si) | O(ti) \equiv \hat{}Y1\}$. To avoid entering into a deadlock, transitions terminated at a deadlock state (i.e., state with no fanout) are excluded from $C(\hat{}si)$. Four distinct scenarios are considered for the determination of $\hat{}ti$.
1) *Case 1:* there is only one output compatible transition, $|C(\hat{}si)| = 1$, then $\hat{}ti = C(\hat{}si)$ and $\hat{}si+1 = D(\hat{}ti)$.
2) *Case 2:* if more than one output compatible transition are found, i.e., $|C(\hat{}si)| > 1$, then a transition from $C(\hat{}si)$, with the next state having the highest number of free input combinations, will be selected as$\hat{}ti$. Its output will be modified to $O(\hat{}ti) = O(\hat{}ti) \cup \hat{}Yi$ and $\hat{}si+1 = D(\hat{}ti)$.
3) *Case 3:* if $|C(\hat{}s_i)| = 0$, then the free input combinations of $\hat{}s_i$ will be considered. Let $F(\hat{}s_i) = \{X \in \_ | \delta(\hat{}s_i, X) = \emptyset\}$ be the set of free input combinations of $\hat{}s_i$. For $F(\hat{}s_i) \_= \emptyset$, let $D(\hat{}s_i) = \{\hat{}s_j \in Q | \hat{}s_j = D(\hat{}t_i)\ \forall \hat{}t_i \in T(\hat{}s_i)\}$ 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 $D(\hat{s}_i) = \emptyset$. When $D(\hat{s}_i) = \emptyset$, $\hat{s}_{i+1}$ 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.

4) *Case 4:* if $|C(\hat{s}i)| = 0$ and $F(\hat{s}i) = \emptyset$, then a pseudo input variable $xn+1$ needs to be introduced in M and the number of input variables n is incremented by 1. $xn+1$ 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 $D(\hat{s}i) = \emptyset$, 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 $I(\hat{t}i)$.

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 $O(\hat{t}i) = l[\hat{s}i, I(\hat{t}i)] = \hat{Y}i$ 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, $X \in I(tu)$, $tu \in T(\hat{s}i)$, 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."

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 $2n-1$ potential free input combinations are created in every state transition, where n 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.

```
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(ŝ_i → 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.

```
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.

FSM_watermarking $(M, MSG, m, k, N, Q, K_e)$ {
  $W = \{w_i\}_{i=1}^{m} =$ **MD(encrypt**(MSG, K_e)**)**;
  $B = \{b_i\}_{i=1}^{m} = \text{PNG}(K_e, N \times k)$, $b_i \in [1, N \times k]$;
  $\hat{\mathbf{Y}}$ = **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.

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.

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 are long. 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 *Awmi* denote the area of watermarked FSM with $N = Ni$ at the *i*th trial. $Ni = Ni{-}1 \bullet \} \ \delta i$ and $N1 \approx m$. *Ni* that is incremented (or decremented) by δi depends on the extent to which $Awmi{-}1$ is increased (or reduced) over the previous trial. The standard deviation, σi, of *Awm* is defined as

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

N_adaptation $(A, \varepsilon, N_{max}, m)$ {
  $i = 1$; $N_i \approx m$; $A_{wm_0} = A$; $N = N_i$;
  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, $\hat{}\mathbf{X} = \{\hat{}X1, \hat{}X2, \bullet \ \bullet \ , \hat{}XN\}$, applied on state $\hat{}s1$. 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. 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] and [17] to boost the confidence in positive watermark identification. By watermarking the scan chain of watermarked FSM using the techniques proposed in [16] and [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.

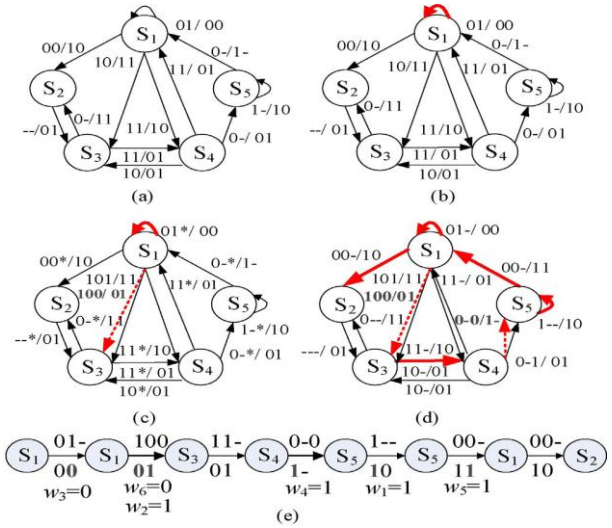

Fig. 7. Example of watermarking on FSM. (a) Original FSM. (b) Use of existing transition. (c) Introduction of pseudo input variable and new transition. (d) Watermarked FSM. (e) Excitation of watermarked transitions.

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 $Q = \{s1, s2, s3, s4, s5\}$. 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 $B = \{9, 4, 2, 7, 12, 3\}$.

Following the algorithm in Fig. 2, since $2(1-1)+1 = 1 / \in B$, $\hat{}y1,1 =$ "–;" since $2(1-1) + 2 = 2 = b3$, $\hat{}y1,2 = w3 =$ "0;" $3 = b6 \Rightarrow \hat{}y2,1 = w6 =$ "0;" $4 = b2 \Rightarrow \hat{}y2,2 = w2 =$ "1;" $5 / \in B \Rightarrow \hat{}y3,1 =$ "–;" $6 / \in B \Rightarrow \hat{}y3,2 =$ "–;" $7 = b4 \Rightarrow \hat{}y4,1 = w4 =$ "1;" $8 / \in B \Rightarrow \hat{}y4,2 =$ "–;" $9 = b1 \Rightarrow \hat{}y5,1 = w1 =$ "1;" $10 / \in B \Rightarrow \hat{}y5,2 =$ "–;" $11 / \in B \Rightarrow \hat{}y6,1 =$ "–;" $12 = b5 \Rightarrow \hat{}y6,2 = w5 =$ "1;" $13 / \in B \Rightarrow \hat{}y7,1 =$ "–" and $14 / \in B \Rightarrow \hat{}y7,2 =$ "–." Hence, $\hat{}\mathbf{Y} =$ "–0 01– –1–1– –1– –."

To verify the existence of watermark W, an input sequence, $\hat{}X =$ ("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 ($N = N$) consecutive transitions through the reachable states of a FSM with k output variables. The output sequence of length N (each output alphabet has k binary bits) will be one of $2k \times N$ possible solutions. The odds that the output sequence contains the identical watermark bits at the positions specified by the author's signature are

$$P_c = \frac{2^{k \times N' - m}}{2^{k \times N'}} = \frac{1}{2^m}. \qquad (2)$$

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 $NC(\tau)$ output sequences detected with at least τ fraction of matched watermark bits when NT random input sequences are applied, then the false positive rate is determined as

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

Where $0 \le \tau \le 1$. 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, $P\lambda$ 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 $P\lambda$ to correct the partially corrupted output subsequence due to tampering.

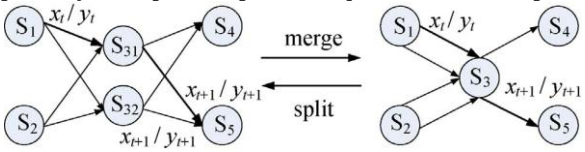

Fig. 8. FSM retiming.

Pc and $P\lambda$ 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 Pc is very low and $P\lambda$ is low enough for a sufficiently large number of random tests.

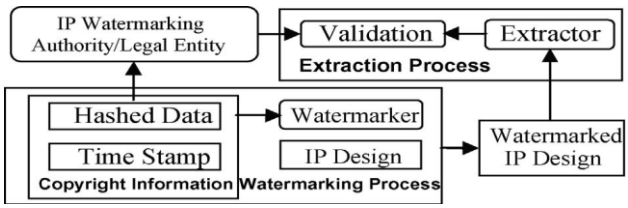

Fig. 9. Watermarking with third party keeping a time-stamped signature.

V. Conclusion

This paper presented 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 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. 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 ultrahigh speed designs that do not have a 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.

VI. REFERENCES

[1] Intellectual Property Protection Development Working Group, *Intellectual Property Protection: Schemes, Alternatives and Discussion*. VSI Alliance, Aug. 2001, white paper, version 1.1.

[2] I. Hong and M. Potkonjak, "Techniques for intellectual property protection of DSP designs," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, vol. 5. May 1998, pp. 3133–3136.

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

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

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

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

[7] A. Cui and C. H. Chang, "Watermarking for IP protection through template substitution at logic synthesis level," in *Proc. IEEE Int. Symp. Circuits Syst.*, 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, "Constraintbased watermarking techniques for design IP protection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 20, no. 10, pp. 1236– 1252, Oct. 2001.

[9] H. J. Kim, W. H. Mangione-Smith, and M. Potkonjak, "Protecting ownership rights of a lossless image coder through hierarchical atermarking," in *Proc. Workshop Signal Process. Syst.*, Oct. 1998, pp.73–82.