Rewiring for Watermarking Digital Circuits

M. Moiz Khan, Spyros Tragoudas
Department of Electrical & Computer Engineering
Southern Illinois University, Carbondale IL 62901
E-mail:{moiz,spyros}@engr.siu.edu

Abstract

A resynthesis method to protect firm cores or circuit netlist representations is presented. The design is protected by embedding watermark by rewiring circuit with one or more redundancy addition/removal steps. There is no or negligible area or timing overhead and existing test sets may be preserved. We analyze several attack strategies and we experimentally demonstrate that the proof of authorship is guaranteed with very high probability for ISCAS’85 and ITC’99 benchmarks.

1. Introduction

In the past few years, the design and development of integrated circuits has seen the rapid development of System-on-Chip (SoC) design philosophy. SoC design methods are a result of endless attempts by industrial design houses to reduce time to market. SoC designs incorporate re-usable, ready to use & tested circuit cores. Using these cores fully functional systems can be made on a single chip and since these cores have already been fully designed and tested, product development time is reduced considerably. The cores used in SoC designs are the intellectual property (IP) of its developer (Alice) and is given to user (Bob). Alice wants Bob to use IP as a black-box and does not want him to change or manipulate the design and maintain proof of her authorship of the design. This forces her to protect her IP and watermark her design.

Hardware available as intellectual property is traditionally classified in 3 formats, hard, firm and soft IPs. Hard IPs, are designed for a specific process. They come as completely or partially routed physical layouts or as netlist along with placement and routing information. They cannot be modified by the user and are intended for use as provided. Examples are timing critical cores like microprocessors and DSP engines. Soft IPs come in the form of synthesizable HDL code (Verilog or VHDL). Although, they are designed using various libraries and portable to be implemented in variety of processes, soft cores typically violate time to market considerations since the user must synthesize the core to gate level and then optimize the layout. An intermediate form are the firm IPs which come as code and structure or gate level netlist to be placed and routed and has features that depend on the target process hence they are less flexible than soft cores. However, they gives more predictable performance from user’s point of view. In this work we propose a scheme to watermark and protect firm IP cores given as gate level netlist.

A watermark or an identification mark can be permanently embedded in the design, as its integral part. Watermark [1] is a mechanism for identification that is designed to: (a) identify the author, the source, the used tools and techniques and/or recipient of the IP, (b) be nearly invisible to human and machine inspection, (c) be difficult to remove, and (d) be permanently embedded as an integral part of the design. Watermark can be inserted by imposing certain characteristics in the solutions of various optimization problems involved in the design phase. For example, [3] gives algorithms to watermark solutions to graph coloring and to final layout [2]. Fingerprinting [4, 7, 8] is a special case of watermarking targeting primarily FPGA designs. A design is partitioned into several tiles and a database with several instances of each tile with fixed peripheral interconnects is generated, the recipient’s signature is inserted in unused CLB of each tile. The final design with different instance of each tile is distributed to different users and the author keeps a record of each recipient to prove his ownership. [5, 6] provide watermarking techniques for soft cores which are typically inapplicable to firm cores.

In this paper we propose a rewiring based watermarking scheme where one or few nodes in the circuit are rewired which are selected based on a signature [14, 15] which identifies the author. Since rewiring may affect the test set and timing properties of the circuit, we do not rewire nodes which affect the test set and timing. In such cases watermarking procedure is repeated with a new signature. Experimental results show that there is space to repeat and get an acceptable signature such that test set and timing are not...
violated. We assume that Alice has the tools and can spend time to optimize the design till an acceptable signature and circuit quality is obtained since this will eventually pay off by prohibiting illegal use of her IP. Section 4 shows that circuit resynthesis by Bob for watermark deletion will result in a task of difficulty equal to complete circuit optimization which is assumed to be prohibitive according to the Strawman proposal [13].

Watermarking combinational logic synthesis solutions have been proposed by Cong et al. [12], where constraints are also imposed on specification by assigning certain nodes to be pseudo-primary outputs. This specification is used as input to the synthesis tool, making the selected nodes to have outputs visible in the final library binding and technology mapping solutions. Our approach is also at the combinational logic synthesis level. However, hardware overhead indicated in [12] is of the order of 10-30% when large number of contraints are used. In contrast, our approach guarantees very low reverse engineering probability without area or timing overhead. The presented method may be combined with that in [12] to strengthen the watermark. [11] embeds watermark on sequential machines presented as firm IPs by manipulating state transition graphs. The approach presented here for combinational circuits is orthogonal to the efforts in [11] and potential benefits by combining our scheme will be investigated in future work. However, in case of full-scan designs protecting the combinational portion of design by our scheme will effectively protect the whole sequential machine.

Section 2 reviews the addition/removal technique of [9] which is the basis of this work. Our watermarking method is given in Section 3. In Section 4 we discuss proof of authorship & possible attacks and Section 5 gives experimental results. We conclude in Section 6.

2. Redundancy Addition/Removal

Redundancy addition/removal (RAR) schemes are ATPG-based resynthesis techniques for circuit optimization. [9] is widely accepted and forms the basis of our approach which we will call CE-RAR. CE-RAR processes all nodes one at a time. Once at a node say $D_i$, it adds a redundant connection in order to remove many newly created ones to optimize logic. The connection $i$ is outside the fanout free cone of $D_i$ and the generated redundant connections are in the fanout free cone as shown in Fig. 2. The problem of identifying connections that can be added without affecting the I/O behavior of the circuit is converted into a test generation problem for stuck-at faults and a mandatory assignment based fast procedure is used to identify redundant connections. After each rewiring step the circuit structure changes. Let $C_0$ be the initial circuit structure, and let $C_i$ be the circuit structure after $i$th rewiring step. The algorithm processes $n$ nodes one after another, and the circuit is transformed from initial representation $C_0$ to $C_1, C_2, C_3 ... C_n$. Throughout the paper we will use the term connections, edges, wire and lines interchangeably.

Consider irredundant circuit in Fig.1 from [9]. If a connection from output of $g5$ is added to input of $g9$ (shown as dashed line) then the functionality of circuit is not changed i.e. it is a redundant connection. However, addition of this connection makes another two wires which were originally irredundant to become redundant. These connections and associated logic is removed to reduce logic in the circuit since CE-RAR aims at optimizing logic. As shown in Fig.2 let $i$ be a added redundant connection which created redundant connection $j \in B$ where $B$ is the set of connections which became redundant due to $i$. It is observed that the following two properties hold for CE-RAR:

**Property 1:** For a pair $(i, B)$, the set $B$ is redundant in the presence of connection $i$ and vice-versa.

**Property 2:** For an added redundant connection $i$, with destination $D_i$, the generated redundant connections are in the fanout-free cone of $D_i$.

The following example demonstrates how CE-RAR may have an impact on the fault coverage of an existing test set
3. Watermarking by Rewiring

Let there be \( n \) total nodes and \( e \) total lines in a circuit. Let \( i \) be an added line with destination node denoted by \( D_i \) and \( j \) be a generated redundant line which may be removed. \( D_i \) is the node pivoted by CE-RAR. Let \( e_{con} \in e \) be the lines in fanout-free cone of destination \( D_i \). From Property 2, \( e_{con} \) defines the space of lines where redundancies can be generated due to addition of \( i \). Let \( n_{ext} \in n \) denote the nodes external to the fanout-free cone of \( D_i \) and \( e_{ext} \) be the edges external to the fanout-free cone of \( D_i \). Let \( E_k \) be the event of adding \( k + 1 \) connections (one at a time, in a sequence), and \( P(E_k) \), be the respective probability of guessing event \( E_k \). We define \( a_i \), \( 0 \leq i \leq k + 1 \) as the average number of different circuits that can be generated at the step \( i \) of CE-RAR. Fig. 3 shows how \( a_i \) is computed in a circuit. Let level 0 be the root of the tree, i.e. \( C_0 \). Initially, \( a_0 \) is 4 at level 0, or, equivalently, we have 4 different instances at level 1. Note that they are different. They correspond to four different nodes of tree depicted at the level 1 in Fig. 3. The number in each node shows the number of redundant connections which can be added in the selective circuit. For example the leftmost node at level 1 in tree indicates a circuit to which three redundant connections can be added. Assume that all 12 circuit instances at level 2 of the tree are different. We have 4 nodes at level 1. Therefore, \( a_1 = \frac{12}{4} = 3 \). If 11 of them were different then \( a_1 \) would have been \( \frac{3}{4} \).

We propose to use CE-RAR as a procedure to embed watermark in the circuit. A single CE-RAR transformation is performed with the objective of creating a watermark that identifies the author. Let us assume that using CE-RAR a redundant connection \( i \) is added with destination node \( D_i \), such that it generates a set of redundant connections \( B \) with \( | B | = r \), the cardinality of set \( B \). With set \( B \) removed due to Property 1, \( i \) is not redundant anymore and acts as a watermark. \( B \) is removed only if test set and timing are not violated. The probability that \( k + 1 \) CE-RAR steps can be guessed is:

\[
P(E_k) = \prod_{i=1}^{k+1} \frac{1}{a_{i-1}}
\]

When multiple steps of CE-RAR are performed then probability to guess CE-RAR steps for watermarking is lowered significantly.

The assignment of a unique identification number to each node is very crucial to our scheme and subsequent verification of the presence of our watermark in case of a dispute. Therefore, we begin the watermarking procedure by assigning a unique identification number or label to each gate \( G_i \) from the set \( G \) of gates in the circuit. The identification number \( i \) is selected from a set \( i \in i = \{1...n\} \) of \( n \) successive numbers, where \( n \) is the cardinality of the set \( G \). However, unique node labeling can be a difficult task due to the existence of functionally & topologically equivalent nodes. Therefore, we insist that a unique labeling system exists to label the designs. In [12] a labeling dependent watermarking scheme was presented. Although [12] proposed a heuristic mapping function which utilizes node functional and timing properties to order the nodes in the circuit, it also called for an EDA standard labeling system. Such a scheme can be used to label our design and for the same reasons we insist that an EDA standard for gate/edge labeling be formed.

To watermark Alice performs following tasks:

Creation & Encryption of Signature message: The signature message is basically an arbitrary text string which Alice wants to embed in the circuit as proof of her authorship. Any text can be used to obtain a sequence of 128-bits. The selected signature text message can be encrypted with a private key of a public key cryptosystem [14]. A well known message digest algorithm like MD5 [15] is used to apply one-way hash function to the encrypted message to obtain...

---

By different we mean different realization but equivalent functionality.
a 128-bit sequence. Using a well know one-way hash function makes it easy to prove the authorship later, since it is well documented that computing their inverse is difficult, and therefore will raise no doubts.

Convert message digest to get node labels: Since there are $n$ nodes in circuit, an $m$-modulo operation ($m \leq n$) on the sequence of 128-bits obtained in previous step, yields a node label $N$. The bit string can be broken into $k$ parts and after performing $m$-modulo operation on each part, $k$ node labels can be obtained. Therefore, $k$ nodes are can be pivoted by CE-RAR.

Embedding Signature: Alice’s signature must be embedded into the design. CE-RAR operations are performed to add edge $i$ with node $D_i$ as the destination by pivoting at $k$ nodes. The added redundant connection $i$ generates a set $B$ of redundant connections which are removed if timing and test set are not violated. Once all the above steps are performed watermark is embedded and watermarking phase is complete. However, to achieve acceptable probabilities, preserve circuit timing and test set it may be necessary that watermarking be done with new signature.

4. Proof of Authorship & Possible Attacks

In this section we present possible attacks scenarios and proof of authorship on the proposed scheme. To authenticate the design Alice needs to prove that a given circuit $C_1$ can be obtained from circuit $C_0$ by performing rewiring corresponding to her signature using CE-RAR. If Bob was to delete Alice’s signature he will have to remove the wire, say $i$, added by Alice and therefore to maintain functionality he will have to add the wire $j$ deleted by Alice (which had become redundant) since that was the alternate connection to $i$. This effectively implies that Bob will come back to circuit $C_0$ which Alice had already made public or declared to a third arbitrating party as her own. In any case wire $i$ or $j$, any one will remain in circuit no matter what rewiring Bob does if the functionality of the circuit is to remain same. However, to detect and delete Alice’s signature Bob will have to first guess the node rewired by Alice. Probability arguments show that there is a very low chance of him being able to guess rewirings done by Alice. Therefore, to steal an IP and claim it to be his own Bob may follow two directions: He may deny the existence of a watermark, or he may again resynthesize the circuit by additional rewiring to delete Alice’s signature or embed his signature on top of Alice’s. We distinguish between these two cases below.

Case 1: Attacks based on denial. Bob may claim that no watermark exists and after having seen the design Alice has created her own signature which predicts application of CE-RAR on precise location in circuit. However, such claims cannot be sustained because to do this Alice will have to come up with a signature from node labels, such that after being encrypted and hashed by a one-way hash function will generate specific node labels needed to apply CE-RAR. But, this is not possible given the inherent computational difficulty in inverting one-way hash functions, therefore the design as claimed belongs to Alice. The remaining of the section focuses on the second case.

Case 2: Attacks based on Resynthesis. To claim IP ownership Bob may pursue the following three resynthesis approaches. (i) Bob may manipulate nodes in circuit. For instance, he may duplicate some nodes to increase drive capability of nodes. In such case the node labels change and it will be hard or impossible for Alice to claim her authorship. However, to follow this direction Bob must first know that circuit has been watermarked using a label dependent rewiring scheme and then rewire nodes in the circuit. However, node duplication will affect the timing and test set of the circuit. Therefore, node manipulation will force Bob to check for timing and test set consistency at each such step, which is a tedious process. In case that Bob decides to rewire every node such that the final circuit is different from that of Alice then this will amount to complete circuit
rewiring maintaining timing and test at each rewiring step. This will result in a task of difficulty equal to complete circuit redesign & optimization which is prohibitive [13] and therefore we assume that an attacker will not pursue this. (ii) Bob may sign the design with his own watermark on top of Alice’s. This is actually not impossible to do and nothing stops Bob from signing the design once again. But to add his signature and then claim the design to be his only, he will have to detect and delete Alice’s signature. The probability of him being able to identify and delete Alice’s signature such that the circuit contains only his signature is given by Eq. (1). (iii) In the third scenario Bob may claim that he followed a different path than Alice to come up with connections claimed as watermark by Alice. He may claim that he performed a different sequence of rewiring steps to reach at the $k$ connections which Alice claims to be her signature. As in Section 3 let $E_k$ be the event of adding $k + 1$ connections, and $P(E_k)$ be the respective probability. We have shown that, even under this attack scenario, the probability of obtaining the sequence of events performed by Alice remains $P(E_k)$ and that our watermarking technique preserves the strength of the signature obtaining technique in [14, 15] with probability $P(E_k)$.

5. Experimental results

Our experimental results focus on demonstrating that large number of CE-RAR transformations are possible on benchmarks for watermarking and that the probability of guessing Alice’s watermarking rewirings is very low.

One of the biggest advantage of our watermarking scheme when compared to other schemes proposed in literature is that we guarantee a very strong proof of authorship with small number of simple steps and the negative impact on the circuit parameters is negligible or very low. For example, we get comparable probabilities to [12, 2], with just 4 rewiring steps, whereas [12] processes large number of nodes comparatively.

Our experimental results on large ISCAS'85 combinational circuit benchmarks (rows 2-6) and full-scan versions of optimized large ITC'99 benchmarks (rows 7-13) are listed in Table 1. We obtained four node labels from a generated signature and we experimented with four iterations of CE-RAR. We added an edge and we removed the set $B$ only if timing and test set were not violated or if it produced a small acceptable violation. We assume unit gate delays and the second column in Table 1 gives the circuit delay under unit delay model. We experimented to find $a_i$ for many steps of rewirings and list $a_i$, the average number of different circuits that can be generated by CE-RAR, for stage/level $C_0, C_1, C_2,$ and $C_3$ in Table 1. Columns 4,7,10 & 13 give delay $\delta_i, 0 \leq i \leq 3$, of each circuit after each rewiring iteration. These results indicate that rewirings can be performed without any timing degradation or with negligible degradation. Observe that for first three iterations almost all circuits showed no timing degration. In addition a negligible timing degradation occurred for a few circuits in 4th iteration.

Columns 3,6,9 and 12 in Table 1 show $a_i, 0 \leq i \leq 3$, at level 0, level 1, level 2 and level 3. It indicates that each benchmark has several nodes which can be pivoted for rewiring at all levels, from level 0 ($C_0$) to level 3 ($C_3$).

Columns 5,8,11 & 14 show the probability $P(E_i), 0 \leq i \leq 3$, for each of the four levels or in other words probability after each of the four rewiring steps of CE-RAR. The experiments show that the signatures can be embedded and the proof of authorship remains very high. These results clearly indicate the effectiveness of the presented method. We re-interate that our approach is an alternative to [12]. It is an orthogonal approach and can be coupled with [12] to increase the strength of the watermark. For a given target $P(E_k)$ one can either use the proposed approach or the approach in [12], whichever is faster. We cannot provide however, an experimental comparison to [12] on this matter since we were unable to acquire that tool. Furthermore, we were not able to find industrial circuits/designs that [12] experimented on.

6. Conclusions

We presented a resynthesis scheme for watermarking digital circuits which has practically no impact on circuit quality. It is a first ever attempt to utilize design rewiring for watermarking. The experimental results indicate that the presented approach results into strong authorship and is appropriate for watermarking without degrading area, test set and timing performance. The presented method is an attractive alternative to [12] since it does not cause area overhead. Future work will couple the existing approach with techniques for sequential circuits that have been presented in [11].

References


<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>Orig. Delay $\delta$</th>
<th>Iteration 1 ($C_0$)</th>
<th>Iteration 2 ($C_1$)</th>
<th>Iteration 3 ($C_2$)</th>
<th>Iteration 4 ($C_3$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>c2670</td>
<td>33</td>
<td>123</td>
<td>33</td>
<td>8.1E-3</td>
<td>108</td>
</tr>
<tr>
<td>c3540</td>
<td>48</td>
<td>308</td>
<td>48</td>
<td>3.2E-3</td>
<td>279</td>
</tr>
<tr>
<td>c5315</td>
<td>50</td>
<td>259</td>
<td>50</td>
<td>3.9E-3</td>
<td>224</td>
</tr>
<tr>
<td>c6288</td>
<td>125</td>
<td>323</td>
<td>125</td>
<td>3.1E-3</td>
<td>299</td>
</tr>
<tr>
<td>c7552</td>
<td>44</td>
<td>298</td>
<td>44</td>
<td>1.4E-3</td>
<td>263</td>
</tr>
<tr>
<td>b14-1</td>
<td>4</td>
<td>326</td>
<td>4</td>
<td>3.1E-3</td>
<td>281</td>
</tr>
<tr>
<td>b15</td>
<td>7</td>
<td>417</td>
<td>7</td>
<td>2.4E-3</td>
<td>411</td>
</tr>
<tr>
<td>b17</td>
<td>9</td>
<td>1012</td>
<td>9</td>
<td>9.8E-4</td>
<td>1129</td>
</tr>
<tr>
<td>b18</td>
<td>7</td>
<td>2732</td>
<td>7</td>
<td>3.6E-4</td>
<td>3014</td>
</tr>
<tr>
<td>b20</td>
<td>7</td>
<td>512</td>
<td>7</td>
<td>2.0E-3</td>
<td>468</td>
</tr>
<tr>
<td>b21-1</td>
<td>7</td>
<td>542</td>
<td>7</td>
<td>1.8E-3</td>
<td>511</td>
</tr>
<tr>
<td>b22-1</td>
<td>7</td>
<td>897</td>
<td>7</td>
<td>1.1E-3</td>
<td>915</td>
</tr>
</tbody>
</table>

Table 1. Probabilities $P(E_i)$ and detected $a_i$ for four levels