Optimization-Intensive Watermarking Techniques for Decision Problems

Jennifer L. Wong, *Student Member, IEEE*, Gang Qu, *Member, IEEE*, and Miodrag Potkonjak, *Member, IEEE*

***Abstract*—Recently, a number of watermarking-based intellectual property protection techniques have been proposed. Although they have been applied to different stages in the design process and have a great variety of technical and theoretical features, all of them share two common properties: 1) they are applied solely to optimization problems and 2) do not involve any optimization during the watermarking process. In this paper, we propose the first set of optimization-intensive watermarking techniques for decision problems. In particular, we demonstrate, by example of the Boolean satisfiability (SAT) problem, how one can select a subset of superimposed watermarking constraints so that the uniqueness of the signature and the likelihood of satisfying the satisfiability problem are simultaneously maximized. We have developed three SAT watermarking techniques: adding clauses, deleting literals, and push-out and pull-back. Each technique targets different types of signature-induced constraint superimposition on an instance of the SAT problem. In addition to comprehensive experimental validation, we theoretically analyze the potentials and limitations of the proposed watermarking techniques. Furthermore, we analyze the three proposed optimization-intensive watermarking SAT techniques in terms of their suitability for copy detection.**

***Index Terms*—Boolean functions, design automation, logic design.**

I. Introduction

PROTECTING software from piracy is one of the most crucial issues in computer science. The time-to-market pressure drives intellectual property (**IP**) into the center of several trends sweeping through today's electronic design automation (EDA) and application specific integrated circuits (ASIC) industries. The requirement for the exchange of IP in the design of system-on-chip is well documented. From the IP providers' point of view, there is an urgent need for protection technique(s) to recoup huge research and development investments on their IP and to keep profits beyond the reach of pirates.

Watermarking or data hiding is designed to meet this demand. In essence, watermarking intentionally embeds digital information into the software for purposes such as identification and copyright. Such information could be the author's name, company name or other messages highly related to the owner and/or the legal users of the software. If necessary, this information can be used in court to prove authorship of the software or proof of legal users entitled to distribute copies.

Manuscript received August 7, 1999; revised October 10, 2002. This paper was recommended by Associate Editor R. Gupta.

J. L. Wong and M. Potkonjak are with the Computer Science Department, University of California, Los Angeles, CA 90095-1596 USA (e-mail: jwong@cs.ucla.edu).

G. Qu is with the University of Maryland Institute for Advanced Computer Sciences, College Park, MD 20742 USA.

Digital Object Identifier 10.1109/TCAD.2003.819900

The newly developed *constraint-based* watermarking technique [1] first translates and embeds an IP author's signature into the original optimization problem as additional constraints. Then, the watermarked problem will be solved and the solution remains valid for the initial problem, since all original constraints are met. The authorship is provided by showing that a randomly obtained and functional solution to the initial problem can rarely satisfy all the signature-based extra constraints.

However, there are two factors that limit the usage of this generic technique. First, watermark embedding further constrains the initial problem and we may lose quality in the (watermarked) solution. In fact, what makes optimization problems hard and interesting is finding an *optimal* solution. In most cases, sacrificing the solution's quality for proof of authorship may not be acceptable and this remains as one of the primary reasons that watermarking has not yet been adopted by the industry. Second, this technique cannot be used directly to watermark decision problems because the signature-based extra constraints may make an originally satisfiable problem unsatisfiable. Decision problems, represented by the Boolean SAT problem, play a central role in theoretic computer science and find numerous applications in various fields. For example, SAT is the first computational task shown to be *NP-hard* and appears in many contexts in the field of very large scale integrated (VLSI) computer-aided design (CAD), such as automatic pattern generation, logic verification, timing analysis, delay fault testing, and channel routing. In sum, watermarking techniques that: 1) *keep the degradation of the solution's quality at the minimal level* and 2) *can be applied to protect decision problems* are needed. In this paper, we propose *optimization-intensive* techniques for such needs. The basic idea is to embed a message in an optimal way such that the probability of changing the solution to the decision problem (or degrading the quality of an optimization problem's solution) is minimized.

In the next section, we review the work in relative areas. Then, we propose the optimization-intensive watermarking methodology for protecting decision problems. As an example, we develop three such techniques for the SAT problem. We analyze these optimization techniques and present the experimental results before concluding.

II. Related Work

A watermark is a mark embedded into an object for identification of the owner and has been used extensively to protect digital image, audio, video, and multimedia data. Existing watermarking techniques take advantage of the limitations on human

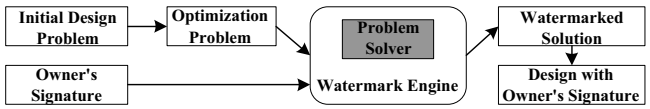

Fig. 1. Illustration of the constraint-based watermarking.

visual and auditory systems and cannot be directly applied for VLSI design IP protection.

The concept of constraint-based watermarking methodology is introduced for the purpose of IP protection in [1]. This generic scheme has been successfully applied at the level of algorithms [2], behavioral synthesis [3], logic synthesis and physical design [1], [4], as well as in field programmable gate array designs [5].

Fig. 1 illustrates the general approach. First, we convert the initial design problem into an optimization problem. Then, we build the watermarking engine that takes the optimization problem and the signature as input and returns a solution with the owner's signature embedded. From this solution, we obtain the watermarked IP.

An effective watermark must provide *high credibility, low overhead, high resilience, perceptual invisibility* and has to be *transparent* to the CAD tools, as well as be able to *protect all parts* of the IP. To apply such technique, we assume that: 1) there exists a well-defined interpretation that maps the IP to solutions of a hard optimization problem; 2) there exist algorithms and/or software packages that solves the optimization problem efficiently; and 3) the optimization problem has a large solution space with acceptable quality to accommodate the owner's watermark.

These assumptions restrict this approach to optimization problems. In this paper, we extend it to the decision problem, SAT. We mention that many heuristics have been developed to solve this problem [6], [7] and rigorous analysis has been conducted based on well-defined random models [8], [9]. The former gives us tools to solve the problem and the latter provides us theoretical background. Most of the current available SAT solvers solve the problem by **systematic search** (such as GRASP [7], ZChaff [10], POSIT [11], NTAB [12], REL_SAT and REL_SAT_rand [13], Satz and Satz-rand [14]); **stochastic local search** (such as GSAT and WalkSat [15]); or by **translating to 0–1 integer programming problems** [15].

III. Constraint-Based Watermarking in an Optimization Fashion

The essence of the constraint-based watermarking method is to cut the solution space by adding extra constraints into the design process of the original IP. This brings us the tradeoff between overhead and credibility (see [16] for a detailed analysis). Briefly, the tighter the extra constraints, the more difficult to solve the optimization problem and, hence, the more degradation of solution quality we may have.

For most optimization problems, we are guaranteed the existence of valid solutions despite their quality. For example, any graph of n vertices is n-colorable in the graph vertex coloring problem. But the decision problems have only two different solutions: YES or NO. For the constraint-based watermarking technique to be effective, we make the following assumption.

Watermarking Assumption

The decision problem to be watermarked must have an answer YES and have many different ways to achieve this answer.

This basic assumption corresponds to the "large solution space" requirement for the constraint-based watermarking on optimization problems. Since the watermarked IP has to maintain the correct functionality, meaning the YES/NO answer in case of the decision problem, the question arises immediately "*Will the YES/NO answer stay unchanged as we watermark the decision problem?*"

It is not hard to construct counter-examples, where we turn a satisfiable SAT formula into unsatisfiable by adding extra constraints, such as additional clauses. In general, adding constraints may cut the solution space. That is, some of the solutions that give the YES answer to the original problem may not make all the extra constraints true and therefore will not be considered as solutions to the watermarked problem. In the worst scenario, the signature-based constraints can make the solution space empty and give NO as the answer to the watermarked problem. The proposed optimization version of the constraint-based watermarking is designed to avoid this by only embedding part of the author's signature.

The idea of optimization-intensive watermarking comes from the following observation. The purpose of a watermark is to provide evidence of authorship. This is achieved by showing the probability that a random generated solution from the initial problem meets all of the signature-based constraints is so small that it is unlikely to happen. An authorship with 100% certainty can never be established, even when all the watermarks are discovered in the IP. So we do not have to embed the entire signature as long as we can provide a convincing proof of authorship.

Specifically, we create a set of constraints from the to-be-embedded watermark. Each constraint makes some solutions invalid, and the constraints do not have the same effect in cutting the solution space. For example, the formula $x_1(x_2+x_3')(x_4+x_5'+x_6)$ can be easily satisfied, and it is still satisfiable after we add new clauses like $x_4(x_2+x_6')(x_1+x_2'+x_4')$, but it becomes unsatisfiable if we add x_1'.

For hard decision problems, there is no simple test that tells us which constraint will cut the solution space slightly and which one may completely change the answer to the problem. In the optimization constraint-based watermarking techniques section we will present in the next section, we intend to add a subset of the constraints from the signature based on certain statistical information while optimally keeping the YES/NO answer to the original decision problem.

IV. Optimization-Intensive Watermarking Techniques

In this section, we present three watermarking techniques on the satisfiability problem to explain our optimization-intensive constraint-based watermarking for decision problems.

Basic Notations:

$\{x_i : i = 1, 2, \ldots, n\}$ is a set of *Boolean variables*, and we denote a variable x_i's *complement* by x_i'.

A *literal* is either a variable or its complement.

Input: a formula $\mathcal{F}$ over boolean variables $\{x_1, \ldots, x_n\}$, integer k, and a message $\mathcal{M}$.
Output: new formula $\mathcal{F}'$ derived from $\mathcal{F}$ with $\mathcal{M}$ embedded.
Algorithm:
1. copy $\mathcal{F}$ to $\mathcal{F}'$;
2. convert $\mathcal{M}$ to a binary string $\mathcal{S}$;
3. while ($\mathcal{S}$ is not empty) {
4. get length l of next clause $\mathcal{C}$ from first k bits of $\mathcal{S}$;
5. construct clause $\mathcal{C}$ from next $l\lfloor 1 + \log_2 n \rfloor$ bits of $\mathcal{S}$;
6. evaluate the objective function Obj at $\mathcal{C}$;
7. if ($Obj(\mathcal{C}) \geq$ preset_threshold) add $\mathcal{C}$ to $\mathcal{F}'$;
8. advance $\mathcal{S}$; }
9. report formula $\mathcal{F}'$;

Fig. 2. Pseudocode for watermarking SAT by adding clauses.

A *clause* is a disjunction (logic-OR, denoted by $+$) of one or more literals. We say a clause is true if and only if at least one of its literals is assigned value 1.

A *formula* is a conjunction (logic-AND, denoted by $\cdot$ or omitted when there is no ambiguity) of one or more clauses. A formula is satisfiable if there is a truth assignment to the variables, such that all of the clauses are true. For example, the formula $\mathcal{F} = x_1(x_2 + x_3')(x_4 + x_5' + x_6)$ over variables $\{x_1, x_2, x_3, x_4, x_5, x_6\}$ is satisfiable and one truth assignment can be $\{x_1 = 1, x_2 = 1, x_3 = \phi, x_4 = \phi, x_5 = 0, x_6 = \phi\}$, where ϕ is *don't-care*, which means the value of this variable does not affect the satisfiability of the given formula.

Finally, for the simplicity of our analysis, we allow *redundancy* in the formula (one variable may appear multiple times in the same clause and a clause can occur in the same formula more than once). For instance, $(x_1)(x_1' + x_2 + x_2')(x_1)$, which is functionally equivalent to x_1 over variables x_1 and x_2, is legal under this notation.

A. Adding Clauses

Given a set of n Boolean variables, there are 2^n truth assignments. This is the potential solution space of any satisfiability problem over this set of variables. A satisfiable formula has a nonempty solution space, while an unsatisfiable formula's solution space is empty. Any clause in a formula is a constraint that will prune the solution space. For instance, clause $x_1 + x_2$ will eliminate all truth assignments that assign both x_1 and x_2 to be 0 and, hence, cut one quarter of the solution space.

In the constraint-based watermarking process, a signature is embedded into the original problem as additional constraints to limit the choice of solutions. The natural constraint in the SAT problem is the clause and, therefore, the most straightforward way to embed signatures is to add new clauses. The extra clauses will be generated from the signature and any watermarked truth assignment will satisfy both the initial clauses as well as these signature-based ones. It is the fact that the additional clauses are met, which is used to prove the existence of the signature. There are various ways to interpret a signature into extra clauses, one is shown in Fig. 2, more details can be found in [3].

What distinguishes this new optimization-intensive watermarking technique from the traditional "blind encoding," which embeds all of the signature, is the selective watermark embedding based on the objective function. The objective function takes clauses as input and returns a nonnegative value, which measures the likelihood that adding these clauses will not change the satisfiability of the formula. We will discuss how to compute the objective function in the next section. As we explained before, it is impossible to construct such an objective function that tells exactly which clauses may change the answer to the formula. We have to test the satisfiability based on the statistic information of the formula.

Input: a formula $\mathcal{F}$ over boolean variables $\{x_1, \ldots, x_n\}$, and a message $\mathcal{M}$.
Output: new formula $\mathcal{F}'$ derived from $\mathcal{F}$ with $\mathcal{M}$ embedded.
Algorithm:
1. convert $\mathcal{M}$ to a binary string $\mathcal{S}$;
2. declare a new formula $\mathcal{F}'$ with no clauses;
3. while ($\mathcal{S}$ is not empty) {
4. get the current clause $\mathcal{C}$ from $\mathcal{F}$;
5. get the next $\lfloor \log_2 l \rfloor$ bits from the current string S, where l is the length of clause $\mathcal{C}$;
6. let k be the number represented in binary by these $\lfloor \log_2 l \rfloor$ bits from $\mathcal{S}$;
7. while ($k > l$) k = next $\lfloor \log_2 l \rfloor$ bits $\mathcal{S}$;
8. delete the k^{th} literal from $\mathcal{C}$ and get $\mathcal{C}'$;
9. evaluate the objective function Obj at clause $\mathcal{C}'$;
10. if ($Obj(\mathcal{C}') \geq$ preset_threshold) append $\mathcal{C}'$ to $\mathcal{F}'$;
11. else append $\mathcal{C}$ to $\mathcal{F}'$;
12. advance both $\mathcal{F}$ and $\mathcal{S}$; }
13. report formula $\mathcal{F}'$;

Fig. 3. Pseudocode for watermarking SAT by deleting literals.

B. Deleting Literals

In general, the longer the clause is, the easier it will be satisfied. (A clause with k literals is false if and only if all k literals are assigned 0). Based on this observation, we propose the second watermarking technique.

For example, let

$$\begin{aligned}\mathcal{F} = &(x_1)(x_3 + x_5)(x_1' + x_5)(x_3 + x_6 + x_7' + x_{10}) \\ &\times (x_2 + x_6' + x_7)(x_1' + x_2' + x_3' + x_4')(x_1' + x_2 + x_5' + x_{10}) \\ &\times (x_1' + x_3' + x_7 + x_8 + x_9')(x_1 + x_5' + x_7) \\ &\times (x_1' + x_2' + x_3 + x_4 + x_6 + x_7' + x_9 + x_{10}').\end{aligned}$$

In this example, the message "*June 1999*" will be embedded, which is 011 011 111 001 111 in binary, where the first four digits represent the month (06) and rest for the year (1999). A nonoptimization version of the above technique, as shown in Fig. 3 without lines 9, 10, and 11, will skip the evaluation of the objective function and simply append every new clause $\mathcal{C}'$ to $\mathcal{F}'$. In this example, literals x_3, x_5, x_7', x_6', x_4', x_{10}, x_1', x_5', and x_{10}' will be deleted, respectively, from $\mathcal{F}$ starting with the second clause

$$\begin{aligned}\mathcal{F}' = &(x_1)(x_5)(x_1')(x_3 + x_6 + x_{10})(x_2 + x_7) \\ &\times (x_1' + x_2' + x_3')(x_1' + x_2 + x_5')(x_3' + x_7 + x_8 + x_9') \\ &\times (x_1 + x_7)(x_1' + x_2' + x_3 + x_4 + x_6 + x_7' + x_9).\end{aligned}$$

Formula $\mathcal{F}'$ has exactly the same number of clauses as $\mathcal{F}$ but with one literal less in each clause (except for single-literal clauses). It is clear that the solution space of $\mathcal{F}'$ is a proper subset of that for $\mathcal{F}$, so any truth assignment that satisfies $\mathcal{F}'$ also satisfies $\mathcal{F}$. However, we see that in this case that $\mathcal{F}'$ is

unsatisfiable because of the single-literal clauses x_1 and x_1'. Therefore, the traditional method fails.

As illustrated in Fig. 3, in the proposed optimization-intensive watermarking process, the strength of each additional constraint is estimated before it is embedded. In this case, for example, it may detect that, after deleting literal x_5 from the third clause $(x_1' + x_5)$, the remaining (single-literal) clause x_1' can hardly be satisfied, i.e., $\mathrm{Obj}(x_1') \ll$ preset_threshold and, thus, the original clause $(x_1' + x_5)$ is kept. For the same reason, the deletion of x_4' from the sixth clause is ignored and we get an optimization-intensive watermarked SAT instance, which is still satisfiable

$$\begin{aligned}\mathcal{F}'' = &(x_1)(x_5)\,(x_1' + x_5)\,(x_3 + x_6 + x_{10})(x_2 + x_7)\\ &\times (x_1' + x_2' + x_3' + x_4')\,(x_1' + x_2 + x_5')\,(x_3' + x_7 + x_8 + x_9')\\ &\times (x_1 + x_7)\,(x_1' + x_2' + x_3 + x_4 + x_6 + x_7' + x_9)\,.\end{aligned}$$

C. Push-out and Pull-back

The constraint-based watermarking techniques add signature-related constraints to the original problem, cut its solution space and, thus, increase the chance of getting a watermarked solution. When these additional constraints are too strong to keep the quality of the solution, we introduce the optimization-intensive technique to embed the constraints in a selective way, which excludes the addition of "bad" constraints. The "adding clauses" and "deleting literals" techniques work on the original solution space to make "good" decisions on whether embedding a constraint or not. They are limited by the size of the solution space and will not perform well on formulas with a small number of solutions. The third technique we propose here breaks this barrier by a two-phase push-out and pull-back procedure.

In the push-out phase, the solution space is enlarged, such that there will be more room to hide the signature. For SAT problem, this can be done by either introducing new variables (and clauses) or deleting clauses. Deletion of clauses may fail to preserve the validity of the solution and, therefore, we focus on introducing new variables. When we treat the SAT instance as a formula over the initial set of variables and a new variable x, the solution space is doubled because x is not involved in the formula and will serve as a "don't care" variable. It is in this larger solution that we apply various (optimization-intensive) watermarking schemes to embed the signature and create a (optimization-intensive) watermarked SAT instance. Once we solve such instance and get a solution over the extended set of variables, we can restrict the truth assignment to the initial variables and the extended solution is pulled back. This is illustrated in Fig. 4 (c) and (d), where the shaded area is the solution space for the watermarked formula.

This technique can be combined with the previous ones and yields a more powerful watermarking method. For example, with the freedom of adding new variables, we can change the "adding clauses" technique in the following way: whenever we detect a dangerous clause, i.e., one that may make the entire formula unsatisfiable, we introduce a new variable to the clause. In this way, we have better chance to maintain the satisfiability of the watermarked formula, and we can build new clauses over the augmented set of variables.

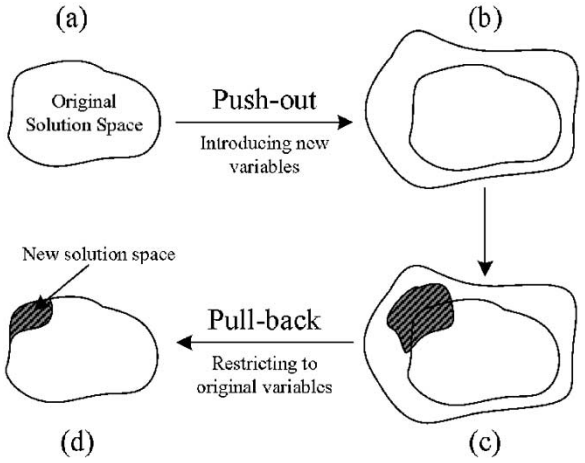

Fig. 4. Illustration of the push-out and pull-back. (a) Solution space for the formula over original variables. (b) Enlarge solution space by introducing new variables. (c) Prune the solution space by embedding watermark. (d) Retrieve solution space for the original formula.

V. Analysis of the Optimization Techniques

In this section, we first show the correctness of the proposed watermarking techniques, then discuss the objective function we mentioned in the previous section. We analyze the limitation of these techniques on one widely-used SAT model and conclude with a discussion on how to detect a watermark from a given solution.

A. Correctness of the Watermarking Techniques

Let $\Psi = \{\mathcal{F} : \mathcal{F}$ is a formula over a set of Boolean variables $\{x_1, x_2, \cdots,\}\}$, we first define a partial order "$\preceq$" on Ψ and say formula $\mathcal{F}_1$ is *more constrained* than $\mathcal{F}_2$, if the partial order $\mathcal{F}_1 \preceq \mathcal{F}_2$ holds.

Definition 5.1: For two clauses $\mathcal{C}_1 = l_{11} + l_{12} + \cdots + l_{1n_1}$, $\mathcal{C}_2 = l_{21} + l_{22} + \cdots + l_{2n_2}$, denote $\mathcal{C}_1 \preceq \mathcal{C}_2$ if and only if $\{l_{11}, l_{12}, \cdots, l_{1n_1}\} \subseteq \{l_{21}, l_{22}, \cdots, l_{2n_2}\}$, i.e., $\forall i \in \{1, 2, \cdots, n_1\}, \exists j \in \{1, 2, \cdots, n_2\}$, such that $l_{1i} = l_{2j}$. For the two formulas $\mathcal{F}_1 = \mathcal{C}_{11} \cdot \mathcal{C}_{12} \cdots \mathcal{C}_{1n_1}$, $\mathcal{F}_2 = \mathcal{C}_{21} \cdot \mathcal{C}_{22} \cdots \mathcal{C}_{2n_2}$, we define $\mathcal{F}_1 \preceq \mathcal{F}_2$ if and only if $\forall j \in \{1, 2, \cdots, n_2\}, \exists i \in \{1, 2, \cdots, n_1\}$, such that $\mathcal{C}_{1i} \preceq \mathcal{C}_{2j}$.

It is clear that the above defines a partial order. Given two formulas $\mathcal{F}$ and $\mathcal{F}'$ with $\mathcal{F} \preceq \mathcal{F}'$, then for every clause $\mathcal{C}'$ (constraints to the SAT instance) in $\mathcal{F}'$, there exists a clause $\mathcal{C}$ in $\mathcal{F}$, such that $\mathcal{C}'$ will be satisfied whenever $\mathcal{C}$ is. For example, $\mathcal{F}$ has all the constraints that $\mathcal{F}'$ has.

When a signature is added as extra clauses, the watermarked formula will become more constrained than the original one and therefore any watermarked solution will remain valid. In summary, we have the following observations.

Proposition 5.2: If $\mathcal{F} \preceq \mathcal{F}'$ and $\mathcal{F}$ is satisfiable, then $\mathcal{F}'$ is satisfiable and any truth assignment to $\mathcal{F}$ satisfies $\mathcal{F}'$.

Proposition 5.3: Let $\mathcal{F}'$ be a (optimization-intensive) watermarked formula from an original formula $\mathcal{F}$, then $\mathcal{F}' \preceq \mathcal{F}$. Hence, any watermarked truth assignment to $\mathcal{F}'$ meets the requirement of $\mathcal{F}$.

B. Objective Function

An objective function $\mathrm{Obj} : \Psi \rightarrow R^+ \cup \{0\}$ measures the likelihood that a formula can be satisfied. Ideally, any objective function Obj should assign unsatisfiable formulas a value of 0 easy SAT instances larger values and be nondecreasing over the partial order $(\Psi, \preceq)$, such as $\mathrm{Obj}(\mathcal{F}) \leq \mathrm{Obj}(\mathcal{F}')$ for any formulas $\mathcal{F} \preceq \mathcal{F}'$. For example, it can be defined as:

- $\mathrm{Obj}(\mathcal{F}) = \min_{1 \leq i \leq n}\{\mathrm{Obj}(\mathcal{C}_i)\}$ for any formula $\mathcal{F} = \mathcal{C}_1 \cdot \mathcal{C}_2 \cdots \mathcal{C}_n$;
- $\mathrm{Obj}(\mathcal{C}) = \Sigma_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{Obj}(\mathrm{l_i})$ for any clause $\mathcal{C} = l_1 + l_2 + \cdots + l_n$;
- extend the notation by denoting $\mathrm{Obj}(l)$ as the likelihood that literal l is assigned true.

The only part left to be specified is how to determine the values of $\mathrm{Obj}(l)$ and $\mathrm{Obj}(l')$ for a literal l and its complementary l'. Intuitively, the more often a literal l appears in the formula and the less its complementary occurs, l will have a better chance to receive true. Let n_l be the number of occurrence of l and we can finish the definition.

Zeroth-Order Objective Function:

$$\mathrm{Obj}_0(l) = \begin{cases} \frac{f_0(l)}{f_0(l')} & n_{l'} \neq 0 \\ \infty & n_{l'} = 0, n_l \neq 0 \\ \text{not defined} & n_{l'} = 0, n_l = 0 \end{cases} \tag{1}$$

where

$$f_0(l) = n_l. \tag{2}$$

Basically, (1) uses the ratio of the literal occurrences as the measurement for the assigning variables true/false. If l's complementary form l' never appears in the formula, to find a truth assignment, it does not hurt us at all to make l true. And if the formula does not contain a particular variable, there is no need to define the objective function on this variable.

First-Order Objective Function: From the zeroth-order objective function, we see that every occurrence of l will increase $\mathrm{Obj}(l)$ and decrease $\mathrm{Obj}(l')$. However, the contribution of each occurrence is related to the length of the clause and this is not considered in the zeroth-order objective function. The literal in any single-literal clause has to be assigned true while any single literal in a clause with many literals is not crucial to the satisfiability of that clause. More specifically, let n_l be the number of clauses that contains the literal l and s_i be the length of the i^{th} such clause. Then, we define the first-order objective function on l as

$$\mathrm{Obj}_1(l) = \begin{cases} \frac{f_1(l)}{f_1(l')} & n_{l'} \neq 0 \text{ and } f_1(l') \neq \infty \\ \infty & n_{l'} = 0, n_l \neq 0 \\ \text{not defined} & n_{l'} = n_l = 0 \\ & \text{or } f_1(l) = f_1(l') = \infty \end{cases} \tag{3}$$

where

$$f_1(l) = \sum_{i=1}^{n_l} \frac{2^{s_i - 2}}{2^{s_i - 1} - 1}. \tag{4}$$

There are $2^k - 1$ distinct truth assignments for a clause of length k, if the clause contains no two literals which are complements of each other, out of which 2^{k-1} will have a particular literal assigned true. Equation (4) is a simple modification of this fact which enforces f_1 to evaluate to ∞ at literals from the single-literal clauses. From this definition, it is easy to verify the following.

Proposition 5.2: The first-order objective function satisfies:

i) $\mathrm{Obj}(l) = \infty$ iff the formula does not contain l' or has l, but not l', as a single-literal clause.
ii) $\mathrm{Obj}(l)$ is increasing with respect to n_l and decreasing with respect to $n_{l'}$.
iii) $\mathrm{Obj}(l)$ is decreasing with respect to s_i.

Case i implies that if the formula does not have l' or has l as a single-literal clause, then setting l true only helps us finding a solution. When the formula has both l and l' as single-literal clauses, obviously it is unsatisfiable. Case ii suggests that the more l occurs, the more likely it will be assigned true, and Case iii says the longer is the clause, the less it contributes to the objective function since a long clause is easier to satisfy.

Second-Order Objective Function: Although the function Obj_1 is better than Obj_0 in describing the likelihood of a literal being assigned true, by no means is it the most accurate. Considering the two clauses $\mathcal{C}_1 = l_1 + l_2 + l_3$ and $\mathcal{C}_2 = l_1 + l_4 + l_5$, $\mathcal{C}_1$ and $\mathcal{C}_2$ will contribute the same amount $2/3$ to $f_1(l_1)$ by (4). However, this becomes inaccurate if we know, from the rest of the formula, that most likely l_2 or l_3 will be true, while both l_4 and l_5 are false. In such a scenario, clause $\mathcal{C}_2$'s satisfiability depends more on the value of the literal l_1 and it should contribute more to the objective function $f_1(l_1)$ than $\mathcal{C}_1$. This suggests us that we should also study the correlation between literals. By modifying $f_1(l)$, we can define the second-order objective function in a similar way with

$$f_2(l) = \sum_{i=1}^{n_l} \frac{1}{2^{s_i - 1} - 1 + \Sigma_{j=1, l_j \neq l}^{s_i} \mathrm{Obj}_1(l_j)}. \tag{5}$$

The purpose of introducing objective functions is to provide criteria that can be used to determine whether an additional constraint should be embedded during the optimization-intensive watermarking process. An objective function estimates the difficulty of determining the satisfiability of a formula. Obj_0 considers only the occurrence of l, l' and uses the ratio as a measure. Obj_1 takes into account the length of each clause that a literal and its complementary appears. In the second-order objective function, not only l and l', but also their *neighbors* (the literals in the same clause) are considered. Therefore, it provides more accurate estimation. Of course, better objective functions can be defined when we use more information from the SAT problem. Unfortunately, since the objective function will be called frequently, the computation cost of such a function should be as low as possible. Usually, the accuracy of the objective function is at the expense of its complexity. For example, both Obj_0 and Obj_1 can be computed when the SAT instance is read in with the help of additional storage. However, one more parse of the SAT instance is required to initialize Obj_2.

To conclude the discussion on the objective function, we mention that a perfect objective function should be able to tell exactly the satisfiability of an instance. However, such function cannot be computed in polynomial time unless $P = NP$. For a

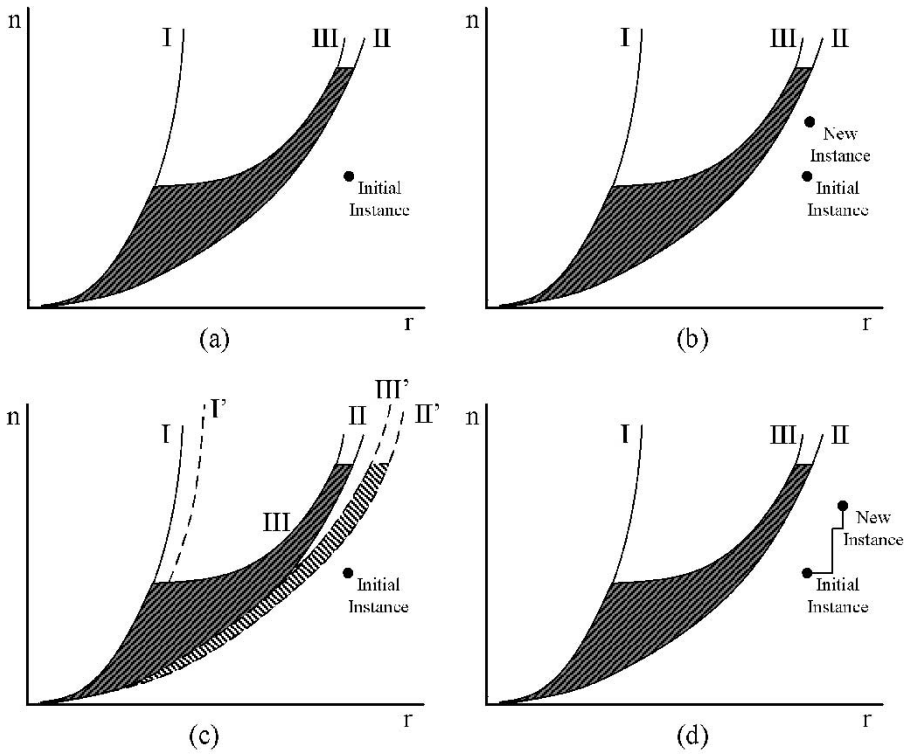

Fig. 5. SAT instance and its watermarked versions. (a) The initial SAT instance. (b) New instance by adding clauses. (c) New instance (same spot as initial) and new curves by deleting literals. (d) New instance by push-out and pull-back.

given satisfiable formula, the optimization watermarking techniques do not guarantee the watermarked formula still satisfiable, but maximizes this probability.

C. Limitations of the Optimization Techniques

The Constant-Probability SAT Model: We adopt the model $J(n, r, p)$ for generating random SAT instances. A formula of this type consists of n clauses of r variables. A variable v is in the kth clause as an uncomplementary literal with probability p, as a complementary literal with probability p, and the kth clause does not contain variable v with probability $1 - 2p$.

Franco and Ho [8] proved that, for this model, almost all SAT instances can be solved in polynomial time if any of the following conditions hold:

$$p < \frac{0.4 \ln n}{r} \tag{6}$$

$$p > \frac{\ln n}{r} \tag{7}$$

$$p = c\frac{\ln n}{r} \lim_{n,r\to\infty} \frac{n^{1-c}}{r^{1-\varepsilon}} < \infty, (0.4 < c < 1, \varepsilon > 0). \tag{8}$$

It is also shown that almost all of the randomly generated SAT have no solution if

$$p = c\frac{\ln n}{r} \text{ and } \lim_{n,r\to\infty} \frac{n^{1-c}}{r} = \infty \, (0.4 < c < 1). \tag{9}$$

The curves shown in Fig. 5(a) and [8] show the relationships between the parameters of model $J(n, r, p)$ that result in random instances that are always solvable in polynomial time. Curve I represents $e^{pr/0.4}$ and the region to the left of it (6) are instances that are always unsatisfiable due to the large amount of clauses. Curve II's e^{pr} and the region to its right (7) corresponds to instances that are almost always satisfiable. According to (9), the instances above curve III are almost unsatisfiable. The shaded area is a mixture of satisfiable and unsatisfiable problems.

Limitations on the Optimization Techniques: Under the "watermarking assumption," a to-be-watermarked SAT instance belongs to the region to the right of curve II as shown in Fig. 5(a), where the solution space is large. After we embed the signature, the SAT instance and/or the curves may change. We do not want the new instance to fall in the area left of curve I or above curve III, where the probability that the new instance is unsatisfiable is almost 1. Even for a satisfiable watermarked instance in the shaded region, it usually becomes hard to find a truth assignment. We now graphically analyze the impact of the proposed watermarking techniques.

Adding clauses: Assuming the message is random, and the length of a new clause is chosen in accordance with the initial instance, then the watermarked instance is still a random SAT problem of the same type, except that the number of clauses has increased. This is shown in Fig. 5(b), where curves I–III remain the same, the new instance is right above the initial one, which indicates an increment of n with the same r and p. It is clear that if we keep on adding new clauses, the watermarked instance will cross curve II, making the instance hard to solve and eventually becomes unsatisfiable.

Deleting literals: If we delete literals based on a random message, our optimization strategy will keep us from deleting single-literal clauses and eliminating any variable completely

from the formula. Therefore, the new instance will be a formula on the same set of variables with the same number of clauses. In the $r-n$ chart [Fig. 5(c)], the new instance shares the same position as the initial one. However, all of the curves have moved toward right because of the decrement of p due to the deletion of literals. When there are only a few literals left, p will become extremely small and all of the curves will cross the SAT instance and make it unsatisfiable.

Push-out and Pull-back: In this technique, new variables only appear in the clauses corresponding to the signature, so it is not appropriate to use the same model. However, the idea can be illustrated by Fig. 5(d), the initial instance is moving along r-axis as we add new variables, then moving up as we append new clauses. New variables are introduced whenever the new instance moves close to curve II and the addition of a new variable keeps the watermarked formula in the region under the "watermarking assumption." Technically, there is no limitation on this technique, if any number of new variables can be added.

D. Copy Detection

Detecting copies is one of the fundamental problems for distributing IPs among different users. An embedded watermark is useful only if the IP provider can detect it and prove his/her authorship to the third party, which is the sole goal of copy detection. Our key idea used to protect the SAT solution is to prune the solution space based on the signature and then get the solution from this small space. The strength of the authorship depends on the size of the solution space for the watermarked problem relatively to the original one. Here, we outline the approaches to retrieve watermarks embedded by the "adding clauses," similar results hold for the other techniques.

In the "adding clauses" method, the solution is forced to satisfy extra clauses according to the signature. Suppose the signature is translated to k clauses of length $w_1, w_2, \ldots, w_k$, respectively. Let

$$P_k = \Pi_{i=1}^{k} \left(1 - \left(\frac{1}{2}\right)^{w_i}\right). \tag{10}$$

Then, we have the following.

Proposition 5.3: A random assignment makes all k clauses true with a probability P_k, and the probability that it satisfies at least $m < k$ clauses is

$$P_m = P_k \left\{ 1 + \sum_{i_1=1}^{k} \frac{2^{-w_i}}{1-2^{-w_i}} + \sum_{i_1 \neq i_2, i_1, i_2 = 1}^{k} \frac{2^{-(w_{i_1}+w_{i_2})}}{(1-2^{-w_{i_1}})(1-2^{-w_{i_1}})} + \cdots + \sum_{i_j \text{ distinct}, i_1, \ldots, i_{k-m}=1}^{k} \frac{2^{-\Sigma_{j=1}^{k-m} w_{i_j}}}{\Pi_{j=1}^{k-m} 1-2^{-w_{i_j}}} \right\}.$$

Corollary 5.4: For 3-SAT, where all clauses have length 3

$$P_m = \left(\frac{7}{8}\right)^k \sum_{i=0}^{k-m} \binom{k}{m} \left(\frac{1}{7}\right)^i. \tag{11}$$

It is easy to see from the expression of P_m that this probability can be arbitrarily small, when both k and m are large enough. Thus, this method provides high credibility for signatures of large instances. In practice, for a given SAT instance, from the limitation of the technique we can determine the maximal number of constraints we may introduce. Then, according to the level of credibility we want to achieve, we can calculate the minimal constraints we have to add to the original problem and then fine tune the objective function.

VI. Experimental Results

We have implemented our proposed optimization-intensive watermarking techniques and applied them to a set of instances from DIMACS SAT benchmarks [17], the SAT Competition in Beijing, China [18], the Planning set [19], the All Interval Series [20], and the bounded Model Checking set [21].

The DIMACS benchmark suite contains the inductive inference, circuit fault analysis and constant density model sets. The inductive inference instances are generated from the problem of inferring the logic in an 8-input, 1-output "blackbox." The circuit fault analysis set is based on test-pattern instances for single-stuck-at faults, while the constant density model instances are random P-SAT. Two instances are taken from the SAT Competition in Beijing, China. One of the instances is a block world panning problem (Sussman anomaly on three blocks) and the other is a VLSI design for a 2-bit maximizer. The Planning set consists of instances of the seven blocks world planning problem taken from SATPLAN. The All-Interval Series problem is an arithmetic problem which occurs in serial musical composition. Lastly, the bounded Model Checking set is composed of real industrial hardware designs taken form the IBM research lab for verification technologies. More information on these benchmark sets can be found at [6].

We watermark each of these instances using regular techniques without optimization, then apply the optimization-intensive techniques to embed the same message. The message, which we assume is infinitely long, is embedded ten bits at a time. We solve the watermarked formula and embed the next ten bits, if we can find a satisfiable assignment. We stop when the watermark makes the problem instance unsatisfiable and the length of the embedded message measures the power of the watermarking technique. The results show that in most instances, much longer messages can be embedded by the new techniques before changing the problem to unsatisfiable. Both the initial and watermarked instances are solved by WALKSAT, GRASP, zChaff, or satz [6], depending on the type of instance. All instances are solved instantaneously on one of these solvers, and therefore the runtime overhead is negligible. Note that the use of any or all of these solvers on any of these instances has no impact on the quality of the watermarking approach. Some of the instances used to test our approach are solved easier by different types of algorithms, therefore only affecting the runtime of the solvers. For example, the logistics-planning problems are easier to solve with a local search algorithm, while All-Interval Series instances are very difficult for local search algorithms such as WalkSAT, but rather easy for systematic algorithms like satz.

Among the techniques we proposed, the "adding clauses" method has the best performance in terms of the length of the

TABLE I
IMPROVEMENT, IN TERMS OF THE LENGTH OF MESSAGE BEING EMBEDDED, OF THE OPTIMIZATION-INTENSIVE TECHNIQUE OVER REGULAR WATERMARKING TECHNIQUE

Instance	Without Optimization	Optimization Intensive	Improvement Ratio
2bitmax_6	990	1650	66.67%
3blocks	240	540	125%
ais6	580	1000	72.41%
ais8	290	600	106.90%
ais10	310	880	183.87%
ais12	230	540	134.78%
bmc-ibm-2	340	640	88.24%
ii8a1.cnf	1900	1400	-26.32%
ii8a2.cnf	4900	3100	-36.73%
ii8a3.cnf	3900	6700	71.79%
ii8a4.cnf	3900	3900	0%
ii8b1.cnf	3800	6800	78.95%
ii8b2.cnf	4900	8900	81.63%
ii8b3.cnf	4900	8900	81.63%
ii8b4.cnf	3900	11600	197.44%
ii8c1.cnf	3900	7500	92.31%
ii8c2.cnf	8900	13000	46.07%
jnh201	960	1200	25%
jnh209	320	710	121.88%
jnh218	290	650	124.14%
logistics.a	170	270	58.82%
logistics.b	610	840	37.70%
logistics.c	510	750	47.06%
ssa7552-038	2000	3700	85%
ssa7552-043	1320	3100	134.85%
ssa7552-160	1240	4100	230.65%
Average Improvement			85.76%
Median Improvement			81.63%

embedded message before changing the problem to unsatisfiable. We first generate a long random bit-stream as our message, then create clauses of variable lengths according to this message and append them to the original problem. Table I reports the maximal length of the bit-steam that we can take before turning the problem to unsatisfiable. As one can see from Table I, we achieve an average of 85% improvement.

We also test the proposed methods on random 3-SAT instances, where the literal per clause ratio is fixed at 4.25. These instances are in the range of "hard-to-be-solved" [9]. Although all the problems are known to be satisfiable, it is not expected that many satisfying assignments exist. Therefore, the "watermarking assumption" does not hold. When we try to watermark these problems, very limited message can be embedded (less than 100 bits), and the optimization-intensive techniques do not help that much. [Imagine an instance very close to curve II in Fig. 5(a)].

VII. CONCLUSION

Current watermarking techniques can only be used for the protection of IPs which are related to optimization problems. The need for effective methods to protect decision problems is urgent because of their numerous applications in various fields. In this paper, we propose the first set of optimization-intensive watermarking techniques for decision problems. The basic concept of these techniques is to select a subset of the signature and embed it as the watermark. Theoretically, we showed that this partial signature will provide convincing authorship and an average of 85% improvement, in terms of the amount of information embedded, is achieved in practice when we implement this idea to watermark a set of benchmark SAT instances.

REFERENCES

[1] A. B. Kahng, J. Lach, W. H. Magione-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. Computer-Aided Design*, vol. 20, pp. 1236–1252, Oct. 2001.

[2] L. Boney, A. H. Tewfik, and K. N. Hamdy, "Digital watermark for audio signals," in *Proc. Int. Conf. Multimedia Comput. Syst.*, 1996, pp. 473–480.

[3] G. Wolfe, J. L. Wong, and M. Potkonjak, "Watermarking graph partitioning solutions," *IEEE Trans. Computer-Aided Design*, vol. 21, pp. 1196–1204, Oct. 2002.

[4] D. Kirovski, Y.-Y. Hwang, M. Potkonjak, and J. Cong, "Intellectual property protection by watermarking combinational logic synthesis solutions," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, 1998, pp. 194–198.

[5] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "Robust FPGA intellectual property protection through multiple small watermarks," in *Proc. IEEE/ACM Design Automation Conf.*, 1999, pp. 831–836.

[6] The Satisfiability Library, SATLIB. Available: http://www.satlib.org [Online]

[7] J. Marques-Silva and K. Sakallah, "GRASP: A search algorithm for propositional satisfiability," *IEEE Trans. Comput.*, vol. 48, pp. 506–521, May 1999.

[8] K. Franco and K. Ho, "Probabilistic performance of a heuristic for the satisfiability problem," *Discrete Appl. Math. Combinatorial Oper. Res. Comput. Sci.*, vol. 22, pp. 35–51, 1988.

[9] P. Cheeseman, B. Kanefsky, and W. M. Taylor, "Where the really hard problems are," in *Proc. 12th Int. Joint Conf. Artificial Intell.*, 1991, pp. 331–337.

[10] L. Zhang, C. F. Madigan, M. W. Moskewicz, and S. Malik, "Efficient conflict driven learning in Boolean satisfiability solver," in *Proc. Int. Conf. Computer-Aided Design*, 2001, pp. 279–285.

[11] J. W. Freeman, "Improvements to propositional satisfiability search algorithms," Ph.D. dissertation, Comput. Info. Sci., Univ. Pennsylvania, Philadelphia, 1995.

[12] J. M. Crawford, "Solving satisfiability problems using a combination of systematic and local search," in *Proc. 2nd DIMACS Challenge*, 1993, pp. 1–7.

[13] R. J. Bayardo and R. Schrag, "Using CSP look-back techniques to solve exceptionally hard SAT instances," in *Proc. Principles Practice Constraint Program.*, 1996, pp. 46–60.

[14] B. Jurkowiak, C. M. Li, and G. Utard, "Parallelizing SATZ using dynamic workload balancing," in *LICS 2001 Workshop Theory Applicat. Satisfiability Testing*, 2001, pp. 205–211.

[15] B. Selman, H. Kautz, and D. A. McAllester, "Ten challenges in propositional reasoning and search," in *Proc. 15th Int. Joint Conf. Artific. Intell.*, 1997, pp. 50–54.

[16] G. Qu and M. Potkonjak, *Intellectual Property Protection in VLSI Designs: Theory and Practice*. Norwell, MA: Kluwer, 2003.

[17] DIMACS Available: http://dimacs.rutgers.edu [Online]

[18] International Competition and Sympoisium on Satisfiability Testing Available: http://www.cirl.uoregon.edu/crawford/beijing/call.final [Online]

[19] H. Kautz, D. McAllester, and B. Selman, "Encoding plans in propositional logic," in *Proc. Int. Conf. Principles Knowledge Represent. Reasoning*, 1996, pp. 374–384.

[20] H. H. Hoos, "Stochastic Local Search—Methods, Models, Applications," Ph.D. dissertation, Comput. Sci. Dept., Techn. Univ. Darmstadt, Darmstadt, Germany, 1998.

[21] O. Shtrichman, "Tuning SAT checkers for bounded model checking," *Computer-Aided Verification*, pp. 480–494, 2000.

Jennifer L. Wong (S'99) received the M.S. degree in computer science and engineering, in 2002, from the University of California, Los Angeles, where she is currently pursuing the Ph.D. degree.

Her research interests include intellectual property protection, embedded systems, and ad-hoc sensor networks.

Gang Qu (M'03) received the B.S. and M.S. degrees in mathematics from the University of Science and Technology of China, in 1992 and 1994, respectively, and the Ph.D. degree in computer science from the University of California, Los Angeles, in 2000.

He joined the Electrical and Computer Engineering Department, University of Maryland, College Park, in 2000. He became a member of the Institute of Advanced Computer Studies, University of Maryland, in 2001. His research interests include intellectual property reuse and protection, low-power system design, applied cryptography, and computer-aided synthesis.

Prof. Qu has won several awards including the 36th Design Automation Conference Graduate Scholarship and the 2002 George Corcoran Award for Significant Contributions to Electrical and Computer Engineering Education.

Miodrag Potkonjak (S'90–M'91) received the Ph.D. degree in electrical engineering and computer science from the University of California, Berkeley, in 1991.

In 1991, he joined C&C Research Laboratories, NEC USA, Princeton, NJ. Since 1995, he has been with the Computer Science Department, University of California, Los Angeles. His watermarking-based intellectual property protection research formed a basis for the virtual socket initiative alliance standard. His research interests include system design, embedded systems, computational security, and intellectual property protection.

Dr. Potkonjak received the National Scientific Foundation CAREER Award, the OKAWA Foundation Award, the UCLA TRW SEAS Excellence in Teaching Award, and a number of Best Paper Awards.