Watermarking Graph Partitioning Solutions

Greg Wolfe, Jennifer L. Wong, *Student Member, IEEE*, and Miodrag Potkonjak

***Abstract*—The authors introduce an Intellectual Property Protection (IPP) technique for graph partitioning which *watermarks* solutions to the graph partitioning problems so that they carry an author's signature. This technique is completely transparent to the actual computer-aided design tool which does the partitioning and is implemented by preprocessing and postprocessing alone. The authors propose five different schemes for the watermarking of partitioning solutions. The goal is to construct a partitioning solution which not only has a small edge cut, but also encodes the signature of the author. The key idea of all of our schemes is to map the signature into a set of constraints and then satisfy a disproportionate number of these constraints. Four of our schemes are based upon the idea of encouraging groups of vertices to be in the same partition. The fifth is based upon the encouragement of certain edges to be cut by the partitioning. The fifth scheme shows superior performances on all of the cases which we tested, including both two-way and multiway partitioning. The watermarking scheme produces solutions that have very low-quality degradation levels, yet carry signatures that are convincingly unambiguous, extremely unlikely to be present by coincidence and difficult to detect or remove without completely resolving the partitioning problem.**

***Index Terms*—Partitioning, VLSI design, watermarking.**

I. Introduction

THE EXPONENTIAL growth of very large scale integration (VLSI) design integration has a number of major ramifications on the design process and the associated research. We focus on one that is exceptionally important: intellectual property protection (IPP). IPP is becoming more important as information becomes rapidly and easily accessible. In the semiconductor industry, explosive proliferation of reusable core-based designs in particular is motivating a need for effective and efficient IPP schemes and tools. We address this issue in the context of graph partitioning.

Graph partitioning is a critical optimization problem that has many applications, particularly in the semiconductor design process. Higher levels of integration emphasizes a need for even more logical and physical level partitioning. Partitioning is the only synthesis task conducted at all levels of the design process, from the system [10] and behavioral levels to the logic synthesis and physical design levels [1]. Partitioning also plays an important role in design analysis; it is widely studied in the simulation, manufacturing, testing, and emulation literature. Outside of VLSI design, partitioning is a widely used step in many engineering and scientific areas, including parallel programming and database storage.

Manuscript received September 1, 1998; revised June 6, 2001 and April 25, 2002. This paper was recommended by Associate Editor R. Gupta.

The authors are with the University of California, Los Angeles, CA 90095 USA.

Digital Object Identifier 10.1109/TCAD.2002.802277.

We offer an IPP technique for graph partitioning which *watermarks* solutions to graph partitioning problems so that they carry an author's signature. We developed, implemented, and evaluated five different schemes for watermarking partitioning solutions. All of our schemes use the same basic principle of mapping an author's signature into a set of constraints and then modifying the partitioning objective function so that a disproportionate number of these constraints are satisfied. Our technique is completely transparent to the actual computer-aided design (CAD) tool which does the partitioning and is implemented by preprocessing and postprocessing alone.

Watermarked solutions have low-quality degradation, yet carry signatures that are convincingly unambiguous, extremely unlikely to be present by coincidence and difficult to detect or remove without completely resolving the partitioning problem. We now introduce our approach using a small example.

Here we outline an example of watermarking on a graph partitioning instance. Consider the graph, G16, in Fig. 1. This graph has 16 vertices and 31 edges. It was created randomly by specifying that there are 16 vertices and that each potential edge will occur with an independent probability of 0.25. We wish to demonstrate that, for even a graph this small, it is possible to watermark solutions of the graph partitioning problem. We will also show, in general, that the potential for watermarking exists by demonstrating what happens to the number of solutions of various qualities when certain constraints are enforced. For the sake of this example the graph partitioning problem is formally defined below.

Problem: MIN k-WAY BALANCED GRAPH PARTITION
Instance: Graph $G = (V, E)$
Solution: A partition of V into k disjoint sets $F = C_1, \ldots, C_k$ with $|C_i| = |V|/k$ for $i = 1, \ldots, k$.
Measure: The sum of the weight of edges between the disjoint sets, i.e., $\sum_{v_1 \in C_i, v_2 \in C_j, i<j, \{v_1, v_2\} \in E}(1)$.
This measure is called the *edge-cut* of the graph.

Variations of this problem allow for weighted vertices or edges, hyperedges rather than edges, and relaxed balance constraints. Finding a solution to any of these problems with minimum edge-cut is NP-hard [9]. For this example, we are concerned with two-way exactly balanced graph partitioning.

The core idea behind our watermarking technique is to select a set of constraints that correspond to our watermark and then to find a solution to the problem that satisfies a disproportionate number of these constraints. We can accomplish this by preprocessing the problem instance and then running the partitioner (any partitioner) on the modified instance. The number

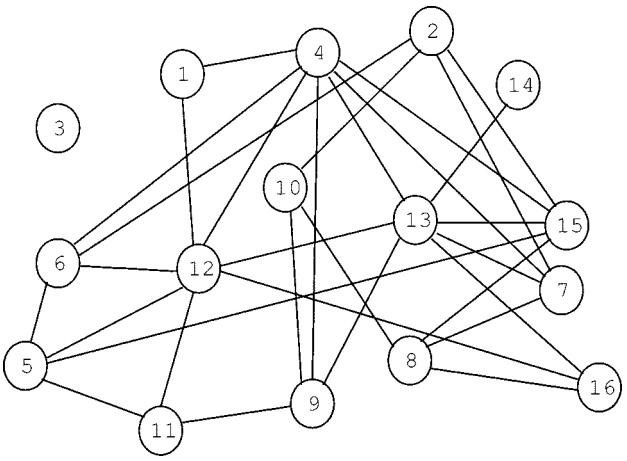

Fig. 1. Flow of the generic forensic engineering approach.

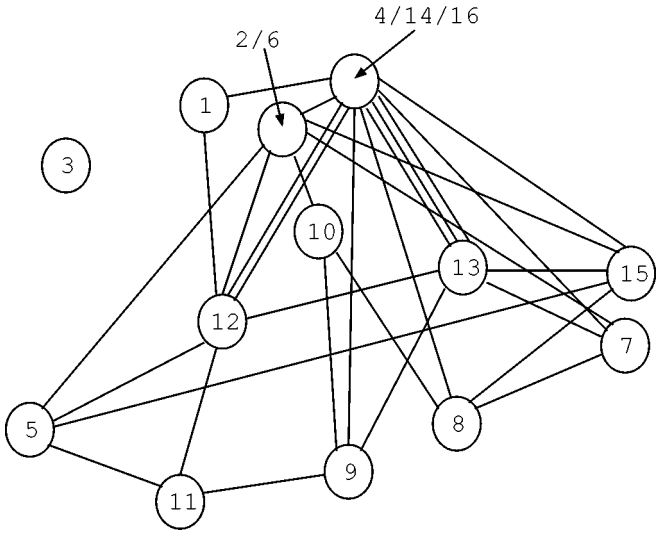

Fig. 2. Graph resulting from G16 after merging the vertex pairs (16,14), (6,2), and (16,4).

and type of constraints imposed, as well as the number that are satisfied, determine the strength of the watermark.

For this illustrative example, our constraints will be of the type "vertex v_1 and vertex v_2 must be on the same side of the partition." We will enforce this by merging the selected vertices as a preprocessing step. Here constraints can never be broken, so we face a simple tradeoff between the number of constraints added and the quality of our solutions. If, for each constraint, v_1 and v_2 are simply selected randomly from the set of vertices, then the probability of each constraint having occurred in some solution by coincidence alone is 1/2. Since these probabilities are independent of each other, the probability of X constraints all occurring in a solution by coincidence is $1/2^X$. Random constraints do not encode a signature, but pseudorandom constraints have the same statistical properties and can encode a signature.

Most partitioning heuristics will quickly find the optimal solution of a problem instance this small, so rather than report the output of an actual partitioner, we will display an exhaustive list of the number of solutions of various qualities in Fig. 3. We do not count partitions that are mirror images of partitions we have already listed. The vertical axis represents edge-cut values and the horizontal axis represents the number of solutions on a logarithmic scale. The outermost curve shows the number of solutions that have various edge-cuts. From this curve one can see that the min cut is 9, that the max cut is 25, and that (for example) there are 371 distinct solutions with an edge-cut of 13. It is important to observe that for this graph there is only one solution with a cut of the minimum size 9. Thus, it is unreasonable to expect to find a watermarked solution which also has cut value of 9.

The other curves correspond to the results of progressively merging pair after pair of vertices together. Eventually, enough vertices are merged together that it is impossible to find balanced solutions at all. This occurred when 12 constraints were enforced. These pairs encode a signature. They were selected using a cryptographically strong pseudorandom number generator seeded in a way that will be discussed later. After the first three of these pairs are contracted (Fig. 2), the min cut is 10, the max cut is 23, and there are 37 different solutions with an edge-cut of 13. All of these solutions satisfy all three constraints and hence there is at most a one-in-eight chance of finding one of these solutions by coincidence. Hence, a partitioner that returned the partition with an edge-cut of 10 would yield a high quality, watermarked solution.

Fig. 4 shows a similar graph for a different list of vertex pairs. It is clear that with the addition of each constraint of this type, the number of solutions of various qualities that satisfy all of the constraints drop by about 50%. This tends to occur evenly throughout the distribution of edge-cut values. As the number of enforced constraints increases, vertices tend to group together into clusters that become large enough to have serious effects on the balance criteria for partitioning. As a result, the drop slowly grows larger than 50%. This does not normally become a concern until a relatively large number of these constraints have been imposed.

The tradeoff between watermarking quality (the strength of the proof of authorship) and design quality is a critical one. In principle, there are two ways how one can discuss this problem. One is mathematically, either combinatorial or probabilistic. Combinatorial analysis is impractical for all but very small examples. For probabilistic analysis, one has to assume a particular distribution for edge existence in a particular type of graph and by using probabilistic methods to calculate the expected results before and after the addition of watermarking constraints. This technique has been very popular in discrete mathematics and in particular the computer science community [3]. The second option is to study the performance of the techniques on a number of real-life designs.

We opted for the second option for two reasons. The first is that to the best of our knowledge, there is no known probabilistic approach for treating the partitioning problem. More importantly, it is well known that different algorithms perform better on different instances of the problem [26]. Therefore, the real test of the effectiveness of the approach and algorithms are instances of real-life standard benchmarks.

II. Related Work

This section reviews related work in the areas of graph partitioning, watermarking, and cryptography.

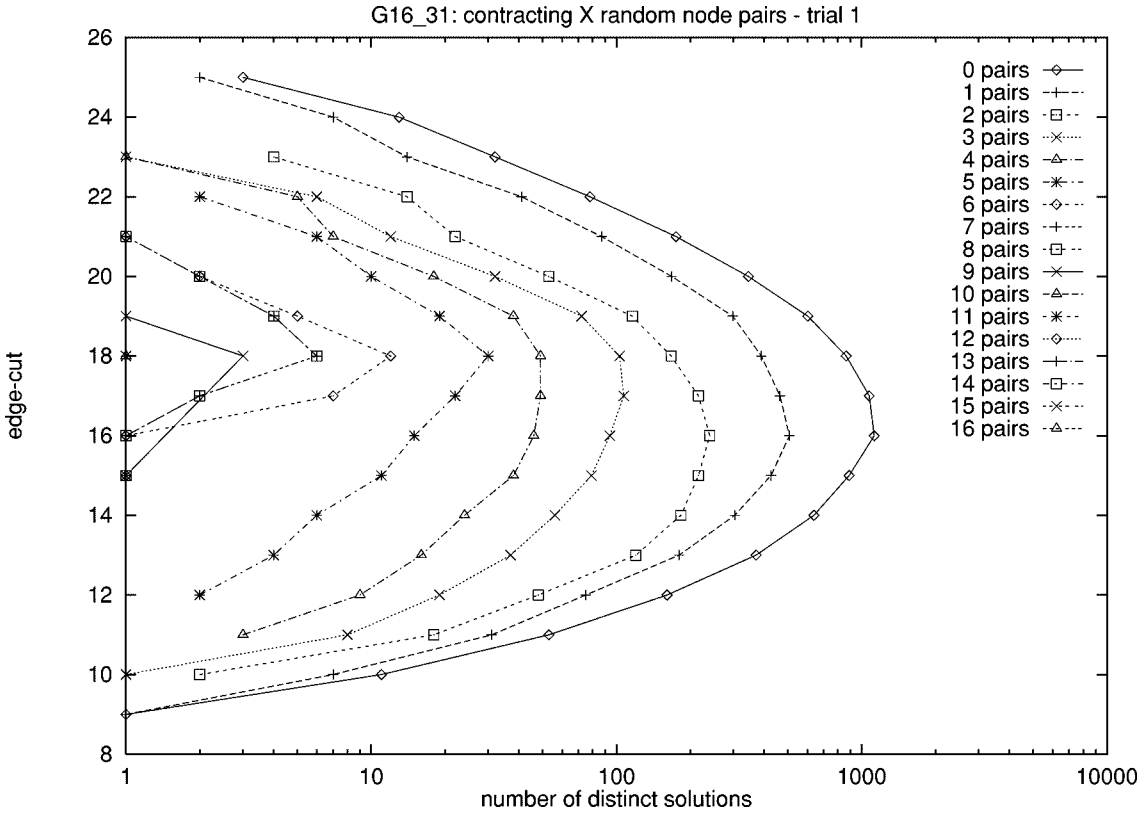

Fig. 3. Number of distinct partitioning solutions of the graph G16 with particular edge-cuts as the following pairs of vertices are merged together (in order): (16,14), (6,2), (16,4), (9,8), (5,16), (9,4), (11,10), (9,4), (15,16), (9,7), (2,3), (13,5), (13,14), (10,12), (14,3), (9,8).

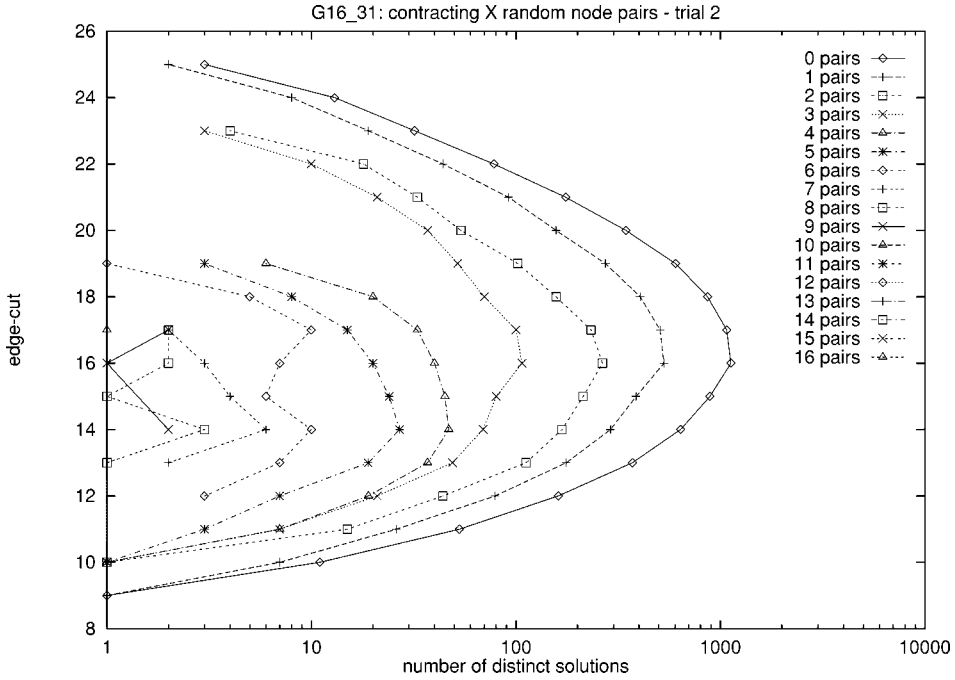

Fig. 4. Number of distinct partitioning solutions of the graph G16 with particular edge-cuts as pairs of vertices are merged together (in order): (7,14), (2,6), (14,9), (13,7), (4,15), (1,13), (11,14), (12,3), (16,15), (13,2), (1,4), (2,3), (12,14), (14,3), (6,12), (6,16).

The graph partitioning problem is ubiquitous in many fields of computer science and engineering. It has important applications in areas ranging from work-load balancing in parallel programming to database storage and in particular to VLSI design and CAD techniques [1].

The graph partitioning problem is NP-complete. Therefore, many heuristic methods are proposed to find high quality partitions. Alpert and Kahng [1] provide a thorough survey of partitioning for VLSI applications. In this paper, we watermark partitions produced by the circuit partitioner of Alpert *et al.*

[2]. We also watermark partitions produced by METIS, a partitioner by Karypis and Kumar [17], which is used primarily for parallel programming applications.

Recently, the development of various IPP techniques such as watermarking [25], [37], fingerprinting [4], [6], [29], forensic engineering [19], metering [21], and obfuscation have attracted a great deal of attention. There are two types of artifacts that can be watermarked: static and active. Watermarking techniques have been proposed for watermarking images [36], video, audio [20], as well as textual objects [5]. Techniques for watermarking computer generated objects such as graphics have been proposed for modeling and animation [32].

For functional artifacts, watermarking techniques have been developed on several levels of abstraction. These levels include system-level design, high level logic synthesis, and physical design [12], [15], [18], [22], [23]. In addition, several new watermarking techniques have been developed. They include multiple watermarking [22], fragile watermarks [24], plagiarism, and software watermarking [7].

Steganography is closely related to cryptography, which has generally received more attention [14]. The traditional task of cryptography is the enciphering of messages using secret codes such that access to the message is limited to those who know how to decipher the message.

Several cryptographic techniques are relevant to the first step of our watermarking approach, specifically, techniques which find a set of constraints to add to the instance of the optimization problem which is to be watermarked. The specific method we use involves the cryptographic hash function MD5, the public-key cryptosystem RSA, and a stream cipher which may be equivalent to the stream cipher RC4 [31]. We use the PGP software package [27] for MD5 and RSA calculations.

Many other cryptographic techniques can potentially be used to introduce important concepts into our watermarking technique. Specifically, digital signatures, revocable signatures, group signatures, undeniable signatures, and zero-knowledge proof signatures have promising applications for watermarking [31].

III. Objectives and Metrics

This section outlines the objectives of watermarking techniques in general and presents metrics with which to measure, compare, and evaluate watermarking techniques on various instances of optimization problems, including those of graph partitioning problems.

The objectives of watermarking techniques vary wildly depending upon the goals of the watermark itself and upon the type of host media that is being watermarked. We assume that the core goal of watermarking is to embed information about the author or owner into an instance of media. We often refer to this information as the signature.

Low Overhead: The watermark should have limited impact upon the quality of the instance of the host media.

Proof of Ownership: The watermark should be convincing evidence to someone who can read it that this instance of the host media was watermarked by the owner.

Hard to Find Ghost Signatures: Ghost Signatures, or more simply *ghosts*, are false signatures or watermarks that are not actually there. An effective watermarking method must assure that the task of finding a ghost signature is either impossible or computationally difficult.

There are many properties of watermarks that may be considered optional. Depending upon the specific goals of a watermark and upon the media the watermarking technique is applied to, each one of these properties may be absolutely necessary, inapplicable, or even undesirable. Our technique satisfies all of the properties listed as follows.

Transparency: The watermarking technique is done in such a way that it is be completely invisible to the other tools used. This allows the watermarking technique to be used in conjunction with any existing or future set of tools that perform the same functions.

Difficult to Detect: The watermark should be difficult to detect. For many types of watermarking techniques, including ours, it may be easy to remove watermarks if they are detected, so difficult to detect is a prerequisite to being tamperproof.

Tamperproof: The watermark is difficult or impossible for an adversary to remove without severe negative impact upon the quality of the watermarked object if the adversary cannot detect the watermark directly, but does know or suspect that a watermark is present.

Difficult to Forge Signatures: It must be difficult for an adversary to place a watermark on an instance of media that claims that the instance belongs to anyone other than the adversary.

The objectives for the watermarking of optimization problems motivates several metrics. These metrics allow the measurement, comparison, and evaluation of watermarks on various problem instances.

Design Metric Degradation: Since solution quality is well defined for optimization problems, the quality degradation can be easily expressed. For minimization problems it is the ratio of the watermarked solution's quality to an unmarked solution's quality minus 1.

Strength of Authorship Proof: The probability, P_c, that a solution to an optimization problem that was not watermarked by a certain author coincidentally contains that author's signature must be low. This clearly must be low so that a competitor's solution is convincingly unlikely to carry your watermark *purely by coincidence*. What should be considered "convincingly unlikely" is very subjective. Probabilities in the range of 10^{-3} to 10^{-12} may arguably be acceptable. If brute force attacks to find ghost signatures are possible, probabilities as low as 2^{-56} may be necessary.

Resiliency Metrics: Attacks that attempt to tamper with a solution until the solution is no longer watermarked usually trade the degradation of the strength of authorship proof with degradation of the solution quality. The strength of authorship proof should be sufficiently strong so that it can degrade somewhat and still be strong enough to be convincing. If the watermarking protocol is such that the possibility of brute force attacks exists, then the strength of authorship proof of a tampered solution should also be strong enough to resist brute force attacks to find ghost signatures.

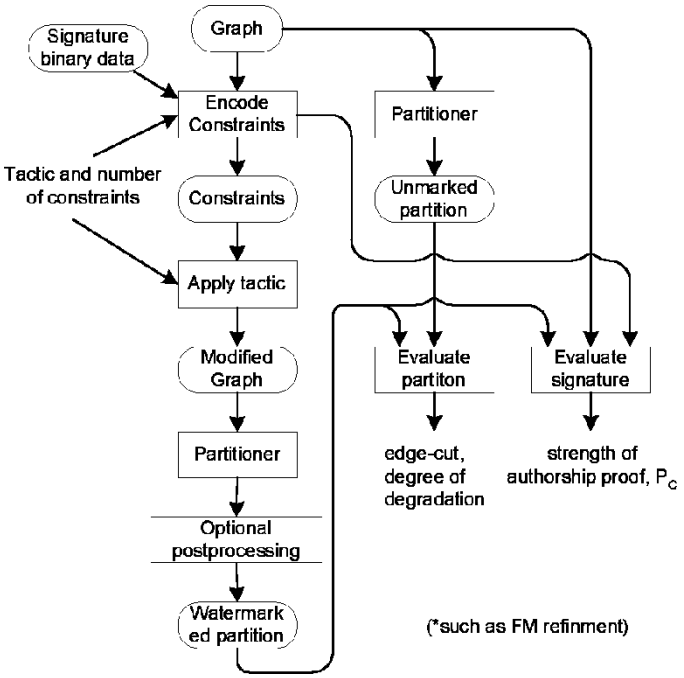

Fig. 5. The watermarking process.

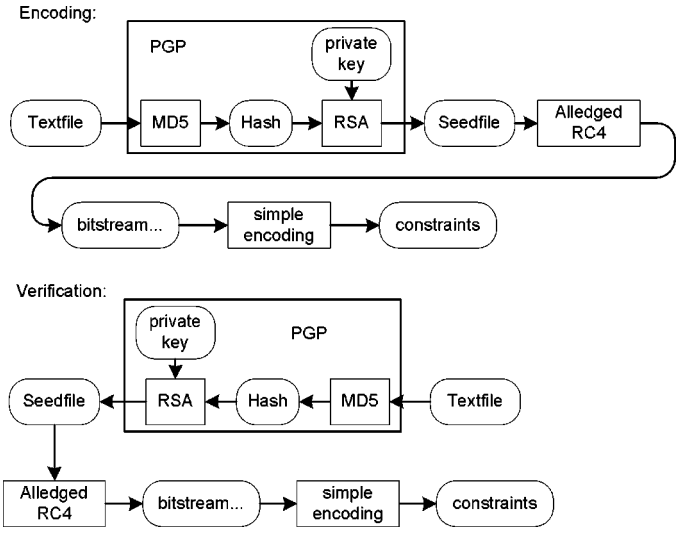

Fig. 6. Process for encoding watermark constraints.

It is also very difficult to predict what type of attacks one can try against the watermarking-based intellectual property protection. Essentially, the question is how much one can benefit from knowing one solution of the computationally intractable problem to generate another sufficiently different solution. Intuition says that the most effective way is most likely to conduct iterative improvement-based search, but other types of attacks can be envisioned. Therefore, one can define resiliency with respect to a set of expected attacks. It is important to emphasize that the resiliency of a watermarked solution is greatly enhanced by the nature of design process. If one alters the solution at one level that almost invariably indicates a need to alter the solution even more at the later levels of the design process. This implies that there is a greatly reduced benefit on conducting any attack.

IV. Approach

The approach for watermarking the solutions of graph partitioning problems is shown in Fig. 5. The general strategy is to add some number of constraints of some type and to satisfy a disproportionate number of them. The type of constraints and the tactics by which they are encouraged to be satisfied are discussed later in this section. All of them have in common the selection of "random" vertices or (hyper)edges. These selections are not truly random and are actually where the user's specific information of the watermark are used. Selections are chosen by a pseudorandom number generator which is seeded by the signature.

The current process we use for generating constraints is shown in Fig. 6. It is done this way so that the encoding scheme yields sufficiently randomized constraints and so that it is difficult to detect and forge signatures.

The figure shows both the encoding process and the process by which one verifies that a signature is present. MD5 is a one-way hash function. RSA is a public key encryption system. "Alleged RC4" is a stream cipher. We use MD5 and RSA only from within the PGP software package. The bit-stream that is the output of "alleged RC4" is a cryptographically strong pseudo-random bit-stream. The "simple encoding" box uses it for tasks like choosing "random" vertices and (hyper)edges.

To verify a signature, one must show that both the signature is present in the partitioning solution and that the signature corresponds to the text file and the PGP public key of the supposed owner. Demonstrating that the signature exists in the partitioning solution is achieved by demonstrating that enough of the signature's constraints are satisfied to be unusual or noncoincidental. One can show that the signature corresponds to the text file and the owner's public key by running PGP.

The protocol for deciding what RSA keys and text files are used is unspecified. If there is any "degree of freedom" in their selection, then a brute force attack may be able to find ghost signatures. In order for this attack to be computationally difficult, P_c must be sufficiently small. $P_c \leq 2^{-56}$ is likely to be strong enough.

Watermarks are added by defining a set of constraints that correspond to the watermark and then finding a solution that satisfies a sufficiently disproportionate number of these constraints. The success of this endeavor can be measured by the amount of design metric degradation and the strength of authorship proof. There are many different types of constraints that can be defined. Additionally, there are many tactics by which these constraints can be imposed in such a way that it is likely that solutions will satisfy a disproportionate number of them. In the introduction, we discussed constraining pairs of vertices to be on the same side of the partition by merging them together. Here, we discuss that tactic as well as four others. We demonstrate their performance in the experimental results section. The choice of which tactic to use and how many constraints to add can make the difference between poor watermarking results and excellent results.

Each tactic uses only preprocessing methods on the problem instance. Postprocessing methods such as FM refinement [8] can be used in conjunction with any of them. The postprocessing

should yield a solution with improved edge-cut (lower design metric degradation), but decreased strength of authorship proof (due to constraints being broken). This may yield a more favorable tradeoff between these two quantities. The constraints imposed by each tactic are completely independent of each other. Because of this, the same constraint may occur several times. In this case it is either satisfied many times or broken many times. Each constraint involves some "random" choice such as choosing a random vertex or edge. As we mentioned before, these choices are not actually random, but use a cryptographically strong pseudorandom number generator.

Because all of the constraints are chosen independently, P_c, the probability of a solution carrying an author's watermark purely by coincidence can be computed by a simple binomial. P_c is a metric for the strength of authorship proof. Let X be the number of constraints imposed, x be the number of these that are *not* satisfied, and p be the probability of a constraint being satisfied purely by coincidence. Note that if constraints were not independent of each other, p would not be well defined. Now, P_c denotes the probability that x or more of the total X constraints are satisfied by coincidence $= \sum_{i=0}^{x} (C_i^X \cdot (p)^{X-i} \cdot (1-p)^i)$.

Overestimating the value p is acceptable, since this will always make P_c larger. A larger value for P_c means a weaker strength of authorship proof, so this can never be used to improve the supposed strength of our watermark. This allows us to estimate p when its exact value is not known. Each tactic depends upon some assumptions. These generally relate to how the graph or hypergraph is specified and to whether the actual partitioner is capable of partitioning certain types of graphs. They are specified with the description of the tactic, but some commonalities are mentioned here.

Watermarking-based IPP relies on the fact that one can establish a one-to-one relationship between text (signature) and the structure of the problem (for the case of partitioning, the relationship between nodes and edges in the graph). Unless the relationship between the signature and the structure (canonical ordering) is established, one can always claim that his text (signature) is actually embedded in a given solution when each constraint is interpreted in a particular way. There are two ways how one can remedy this situation properly. The first relies on the use of one-way functions. A one-way function, $y = f(x)$, is a function where it is easy, in the computational sense, to find y given x, but computationally overwhelmingly difficult and time consuming to calculate x if y is given. Therefore, if it is requested that before embedding the signature one has to go through a one-way function from the x domain being the signature and the y domain being the watermarking constraints, it is clear that it is essentially impossible to find a way of calculating text which corresponds to the given constraints. In practice, to create an effective one-way function it is a standard approach to use seeded pseudorandom generators for this task.

The other option is to use a completely specified canonical ordering of all nodes and edges, and therefore all constraints. For example, one can put the following rank order priority function in place.

1) All nodes with higher degree have smaller number ordering then nodes with smaller degree.
2) If two nodes have the same degree, then the one whose neighbors have a higher total sum of degree has a lower numbering.
3) If two measures are equal, then the node which has the higher sum of squares of the degrees of its neighbors has a lower numbering.
4) If all three previous measures are equal, then the node whose neighbor's neighbors has higher sum of degrees has a lower numbering.

It is obvious that it is easy to derive an arbitrary long priority ranking. Nevertheless, if the graph is highly symmetric this function would not be able to uniquely resolve the numbering of the nodes and edges. While this is extremely unlikely in practice, still it is, at least from the theoretical point of view, safe to use the first option.

In order to watermark, a "canonical" ordering of the vertices should be provided with the graph or hypergraph specification. For all of our tactics we are concerned about the relationships between particular sets of vertices or (hyper)edges and the final partition. In order to properly specify exactly *which* vertices or (hyper)edges we are referring to, a vertex ordering must be provided. Although it is possible to simply choose an arbitrary ordering, it will require solving the graph isomorphism problem to solely determine if we are partitioning the same graph as an attacker who selects a different ordering. Since the graph isomorphism problem is generally considered "hard" [9], this is a bad idea. Hence a "canonical" ordering must be provided. Although it may be possible to have watermarking tactics that do not require this list, all of the tactics listed below do. For similar reasons, some of the tactics also require a "canonical" (hyper)edge ordering as well. For applications like VLSI design where watermarking is likely to occur, the problems are specified very thoroughly and this information is readily available.

Some of the tactics do not apply to certain types of partitioners as they may require that the partitioner function correctly on graphs that have certain properties such as weighted vertices or weighted (hyper)edges. Note that partitioners that can accept hyperedges can automatically accept weighted hyperedges by simply adding duplicates. We introduce each of the tactics below. For each tactic, we discuss the type of constraints, the technique by which they are enforced, the conditions that must exist to use the tactic (assumptions), and the method of computing p, the probability of a constraint being satisfied purely by coincidence. This value p is used to compute P_c.

Merge Random Pairs of Vertices: Random vertices v_1 and v_2 are selected. If they are in the same partition, then the constraint is considered satisfied; otherwise, it is broken. The constraint is imposed by merging the vertices together before the partitioning occurs. The merging process yields a graph that has both weighted vertices and edges. The graph partitioner must be able to partition this type of graph in order for this tactic to be viable. Each constraint is satisfied by coincidence in a two-way partitioning solution with probability 0.5, so $p = 0.5$. For a k-way partitioning solution, each constraint is satisfied by coincidence with probability $p = 1/k$.

Add Edges Between Random Pairs of Vertices: Random vertices v_1 and v_2 are selected. As with the "merge random pairs" tactic, the constraint is considered satisfied only if the two ver-

tices are in the same partition. To make this more likely to occur, an edge is added between the two vertices. If there already is an edge between them, its weight is increased by one. The graph partitioner must be able to partition a graph that has weighted edges for this tactic to work. As above, the constraints are satisfied in a k-way partition by coincidence with probability $p = 1/k$.

Merge Random Edges: Choose a random edge e. The constraint is considered satisfied only if all of the vertices that are incident to the edge are in the same partition. The constraint is imposed by merging all of the vertices together. As with the other merging tactic, the partitioner must be able to partition a graph that has weighted vertices and edges. Additionally, a "canonical" ordering of the edges must be provided so that a random edge may be chosen in a meaningful way.

Let E be the number of edges in the original graph. Let $c(S)$ be the edge-cut of a particular partitioning solution S. Let c be the value of the minimum k-way cut on the graph. Let c' be an estimate to c that is expected to be less than or equal to c. In this paper, we simply guess a "reasonable" value for c', but some methods can produce a true lower bound. Each constraint is satisfied by coincidence in a particular k-way partitioning solution S with probability $p = (E - c(S))/E$. If we wish to provide a single value for p that will not vary with the edge-cut of the solution we are considering, using $p = (E - c')/E$ will work. This will overestimate the true value of p and produce a single conservative value for P_c.

Thicken Random Edges: Choose a random edge e. As with the "merge random edge" tactic, the constraint is satisfied if all of its terminals are in the same partition. This tactic makes this more likely to occur by adding one to the weight of the edge. We refer to this as "thickening" the edge. Using this tactic requires that a "canonical" ordering of edges is provided and that the underlying partitioner can partition graphs with weighted edges. As discussed in merge random edges, the probability of a constraint being satisfied by coincidence is p.

Drop Random Edges: Choose a random edge e. The constraint is considered satisfied if the edge is cut in the partitioning solution (all of its terminals are *not* in the same partition). Otherwise, the constraint is not satisfied. This tactic removes the edge from the graph, so that constraints are more likely to be satisfied. To use this tactic, a "canonical" list of edges must be provided. The host graph partitioner does not need to be capable of partitioning weighted graphs.

Let e be the number of edges in the original graph and let $c(S)$ be the edge-cut of a particular partitioning solution S. Each constraint is satisfied by coincidence in a particular k-way partitioning solution S with probability $p = c(S)/e$. If we wish to compute a single value for p that will protect all solutions whose edge-cut is less than or equal to some value C, then we can set $p = C/e$. Clearly, we must choose a value for C that is greater than or equal to the edge-cut of our watermarked solution.

V. Experimental Results

The key tradeoff of watermarking IPP is between the watermarking quality (degradation of the quality of solution) and the design quality. In this section, we analyze experimentally this tradeoff on a number of popular partitioning design benchmarks. We report watermarking results on several benchmarks from the CAD Benchmarking Laboratory. The benchmarks we report on in this paper are shown in Table I. The first column names the circuits. The second through fourth columns profile the structure of the circuits. Columns five and six show the lowest and highest edge-cut values for which two-way partitions will have a probability P_c of coincidentally containing the watermark that is less than or equal to the one we report. These values are only used in the analysis of some of the tactics. The "merge hyperedges" and "thicken hyperedge" tactics use the "low" values. The "drop hyperedges" tactic uses the "high" value. Columns seven and eight show the same data for four-way partitions.

TABLE I
Benchmark Characteristics

	Description			2-way		4-way	
Graph	# Cells	# Nets	# Pins	low	high	low	high
19ks	2844	3282	10547	100	200	500	650
s13207	8772	8651	20606	50	120	400	700
s15850	10470	10383	24712	40	100	400	700
s35932	18148	17828	48145	35	90	900	1200
s38584	20995	20717	55203	40	100	900	1400
s38417	23849	23843	57613	44	200	800	2000

The underlying partitioner we use is a circuit partitioner by Alpert *et al.* [2]. We experimented with both two-way and four-way partitioning. Fig. 7(a)–(f) shows two-way partitioning. The x axis represents the edge-cut of the watermarked solution. The y axis (scaled logographically) represents the probability P_c of achieving a solution by coincidence. There are five curves, one for each tactic. P_c is computed by a binomial formula as described in the previous section. The binomial formula takes values X, x, and p as inputs. The value p can be computed in an obvious manner. For example, for two-way partitioning on the graph 19 ks, the "merge random pairs of nodes" tactic and the "add hyperedge between random pairs of nodes" tactic both use $p = 0.5$. The "merge random hyperedge" and "thicken random hyperedge" tactics use $p = (3282 - 100)/3282$. The "drop random hyperedge tactic" uses $p = 200/3282$.

The values of X and x are not directly available from the figures, however. As an example, though, consider the point located at about (126, $3 \cdot 10^{-9}$) on Fig. 7(a). This is a point in the middle of the "drop random hyperedges" line. This point corresponds to dropping 300 random hyperedges from the circuit and then partitioning the resultant hyperraph. When this is done, 47 of the constraints are satisfied and 253 are broken. That is 47 of the 155 hyperedges that were cut are from our 300! This is amazing when you consider that the expected value is around 14. Computing the binomial with $X = 300$, $x = 253$, and $p = 200/3282$, we get $P_c = 3.310\,987 \cdot 10^{-09}$. Any partitioning solution of edge-cut 200 or less will have at most this probability of coincidentally containing the watermark. If we had chosen to set $p = 155/3282$ instead, then we would get $P_c = 7.260\,486 \cdot 10^{-13}$, but would only protect solutions whose edge-cut was less than or equal to our own with this strength. Additional experimental results can be found in [35].

Apparent in all of the circuit partitioning experimental results is the superiority of the fifth "drop random edges" tactic. It dis-

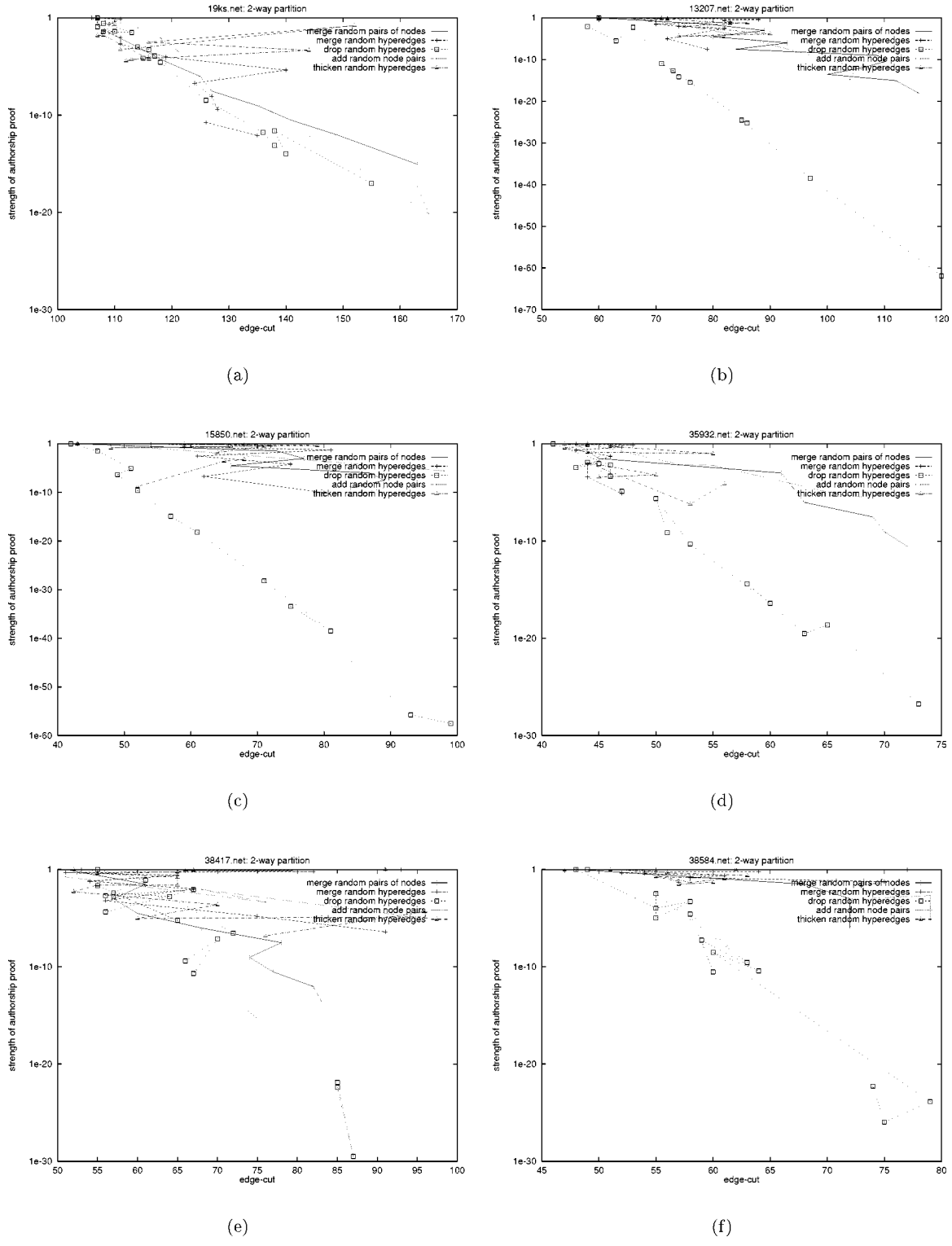

Fig. 7. Tradeoff between degree of edge-cut degradation and strength of authorship proof for two-way partitioning. Five tactics are compared.

plays a very linear pattern on these figures, usually with a slope close to -1. Although the other tactics do occasionally perform better that this one, it is only by a small amount and only for a small region of the graph. This tactic's superior performance stems from a favorable tradeoff between the cost in edge-cut that is paid with each added constraint and the payoff in the strength of authorship proof that is gained with each added constraint.

VI. Conclusion

Partitioning is an ubiquitous task in all synthesis and verification steps of the design process. We proposed the first approach for intellectual property protection of partitioning solutions using a watermarking scheme. We demonstrate the proposed objective function modification watermarking schemes on a

number of real life examples. The technique and our implementation are completely transparent to the CAD synthesis tools and are implemented by preprocessing and postprocessing alone. Solutions produced using our approach simultaneously are very close to the best known solutions, carry signatures that are exceptionally unambiguous, are extremely unlikely to be present by coincidence, and are difficult to detect or remove.

REFERENCES

[1] C. J. Alpert and A. B. Kahng, "Recent directions in netlist partitioning: A survey," *Integration, VLSI J.*, vol. 19, pp. 1–81, 1995.

[2] C. J. Alpert, J. H. Huang, and A. B. Kahng, "Multilevel circuit partitioning," in *Proc. ACM/IEEE Design Automation Conf.*, 1997, pp. 530–533.

[3] B. Bollobas, *Random Graphs*. Orlando, FL: Academic, 1985.

[4] D. Boneh and J. Shaw, "Collusion-secure fingerprinting for digital data," *IEEE Trans. Inform. Theory*, vol. 44, pp. 1897–1905, May 1998.

[5] J. T. Brassil, S. Low, and N. F. Maxemchuk, "Copyright protection for the electronic distribution of text documents," *Proc. IEEE*, vol. 87, pp. 1181–1196, July 1999.

[6] A. E. Caldwell, H. Choi, A. B. Kahng, S. Mantik, M. Potkonjak, G. Qu, and J. L. Wong, "Effective iterative techniques for fingerprinting design IP," in *Proc. ACM/IEEE Design Automation Conf.*, June 1999, pp. 843–848.

[7] C. Collberg and C. Thomborson, "Software watermarking: Models and dynamic embeddings," in *Proc. ACM SIGPLAN-SIGACT Symp. Principles of Programming Languages*, 1999, pp. 311–324.

[8] C. M. Fidduccia and R. M. Mattheyses, "A linear time heuristic for improving network partitions," in *Proc. IEEE Design Automation Conf.*, June 1982, pp. 175–181.

[9] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: W. H. Freeman, 1979.

[10] R. Gupta and G. De Micheli, "Partitioning of functional models of synchronous digital systems," in *Proc. Int. Conf. Computer-Aided Design*, 1990, pp. 216–219.

[11] F. Hartung and M. Kutter, "Multimedia watermarking techniques," *Proc. IEEE*, vol. 87, pp. 1079–1107, July 1999.

[12] I. Hong and M. Potkonjak, "Techniques for intellectual property protection of DSP designs," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, 1998, pp. 3133–3136.

[13] ——, "Behavioral synthesis techniques for intellectual property protection," in *Proc. IEEE Design Automation Conf.*, June 1999, pp. 849–854.

[14] N. F. Johnson, Z. Duric, and S. Jajodia, *Information Hiding: Steganography and Watermarking: Attacks and Countermeasures*. Boston, MA: Kluwer, 2001.

[15] A. B. Kahng, *et al.*, "Constraint-based watermarking techniques for design IP protection," *IEEE Trans. Computer-Aided Design Integrated Circuits Systems*, vol. 20, pp. 776–781, 1236–1252, Oct. 2001.

[16] A. B. Kahng, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe, "Robust IP watermarking methodologies for physical design," in *Proc. ACM/IEEE Design Automation Conf.*, June 1998, pp. 782–787.

[17] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," Dept. Computer Science, Univ. Minnesota, Tech. Rep. TR 95-035, 1995.

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

[19] D. Kirovski, D. Liu, J. L. Wong, and M. Potkonjak, "Forensic engineering techniques for VLSI CAD tools," *Inform. Hiding*, pp. 66–80, 2001.

[20] D. Kirovski and H. S. Malvar, "Robust covert communication over a public audio channel using spread spectrum," *Inform. Hiding*, pp. 354–368, 2001.

[21] F. Koushanfar, G. Qu, and M. Potkonjak, "Intellectual property metering," *Inform. Hiding*, pp. 87–102, 2001.

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

[23] ——, "Fingerprinting techniques for field-programmable gate array intellectual property protection," *IEEE Trans. Computer-Aided Design Integrated Circuits and Systems*, vol. 20, pp. 1253–1261, Oct. 2001.

[24] E. T. Lin and E. J. Delp, "A review of fragile image watermarks," in *Multimedia and Security Workshop*, 1999, pp. 25–29.

[25] S. Meguerdichian and M. Potkonjak, "Watermarking while preserving the critical path," in *Proc. ACM/IEEE Design Automation Conf.*, June 2000, pp. 108–111.

[26] R. Motwani and P. Raghavan, *Randomized Algorithms*. New York: Cambridge Univ. Press, 1995.

[27] P. R. Zimmermann, *The Official PGP User's Guide*. Cambridge, MA: MIT Press, 1995.

[28] G. Qu, J. L. Wong, and M. Potkonjak, "Optimization-intensive watermarking techniques for decision problems," in *Proc. ACM/IEEE Design Automation Conf.*, June 1999, pp. 33–36.

[29] G. Qu and M. Potkonjak, "Fingerprinting intellectual property using constraint-addition," in *Proc. ACM/IEEE Design Automation Conf.*, June 2000, pp. 587–592.

[30] R. L. Rivest, A. Shamir, and L. M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[31] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. New York: Wiley, 1995.

[32] P. Su, J. Kuo, and H. Wang, "Blind digital watermarking for cartoon and map images," *Int. Soc. Opt. Eng.*, vol. 3657, pp. 296–306, 1999.

[33] A. H. Tewfik and M. Swanson, "Data hiding for multimedia personalization, interaction, and protection," *IEEE Signal Processing Mag.*, vol. 14, no. 4, pp. 41–44, 1997.

[34] F. Vahid and D. D. Gajski, "Incremental hardware estimation during hardware/software functional partitioning," *IEEE Trans. VLSI Syst.*, vol. 3, pp. 459–464, Mar. 1995.

[35] G. Wolfe, J. L. Wong, and M. Potkonjak, "Watermarking graph partitioning solutions,", UCLA Tech. Rep. 020 020, May 2002.

[36] R. Wolfgang and E. Delp, "A watermark for digital images," in *Proc. Int. Conf. Images Processing*, 1996, pp. 219–222.

[37] R. Majumdar and J. L. Wong, "Watermarking of SAT using combinatorial isolation lemmas," in *Proc. ACM/IEEE Design Automation Conf.*, June 2001, pp. 480–485.

Greg Wolfe received the B.S. degree in computer science from the University of California, Los Angeles (UCLA). He is currently working toward the Ph.D. degree at UCLA.

His research interests include cryptography, intellectual property protection, combinatorial optimization, and computer games.

Jennifer L. Wong (S'99) received the B.S. degree in computer science and engineering from the University of California, Los Angeles, in 2000. She is currently working toward the Ph.D. degree at the same university.

Her research interests include intellectual property protection and *ad-hoc* sensor networks.

Miodrag Potkonjak received the Ph.D. degree in electrical engineering and computer science from 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 at the University of California, Los Angeles (UCLA). 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 has received the NSF CAREER award, OKAWA foundation award, UCLA TRW SEAS Excellence in Teaching Award, and a number of best paper awards.