Constraint-Based Watermarking Techniques for Design IP Protection

Andrew B. Kahng, John Lach, *Member, IEEE*, William. H. Mangione-Smith, *Member, IEEE*, Stefanus Mantik, *Student Member, IEEE*, Igor L. Markov, Miodrag Potkonjak, *Member, IEEE*, Paul Tucker, Huijuan Wang, and Gregory Wolfe

***Abstract*—Digital system designs are the product of valuable effort and know-how. Their embodiments, from software and hardware description language program down to device-level netlist and mask data, represent carefully guarded intellectual property (IP). Hence, design methodologies based on IP reuse require new mechanisms to protect the rights of IP producers and owners. This paper establishes principles of watermarking-based IP protection, where a watermark is a mechanism for identification that is: 1) nearly invisible to human and machine inspection; 2) difficult to remove; and 3) permanently embedded as an integral part of the design. Watermarking addresses IP protection by tracing unauthorized reuse and making untraceable unauthorized reuse as difficult as recreating given pieces of IP from scratch. We survey related work in cryptography and design methodology, then develop desiderata, metrics, and concrete protocols for constraint-based watermarking at various stages of the very large scale integration (VLSI) design process. In particular, we propose a new preprocessing approach that embeds watermarks as constraints into the input of a black-box design tool and a new postprocessing approach that embeds watermarks as constraints into the output of a black-box design tool. To demonstrate that our protocols can be transparently integrated into existing design flows, we use a testbed of commercial tools for VLSI physical design and embed watermarks into real-world industrial designs. We show that the implementation overhead is low—both in terms of central processing unit time and such standard physical design metrics as wirelength, layout area, number of vias, and routing congestion. We empirically show that in the placement and routing applications considered in our methods achieve strong proofs of authorship are resistant to tampering and do not adversely influence timing.**

***Index Terms*—Intellectual property protection, physical design, VLSI, watermark.**

Manuscript received April 18, 1999; revised November 30, 2000. The work of M. Potkonjak and G. Wolfe was supported in part by the Defense Advanced Research Projects Agency under Grant N66001-97-2-8901. The work of A. B. Kahng, S. Mantik, I. L. Markov, P. Tucker, and H. Wang was supported by a grant from Cadence Design Systems, Inc. This paper was recommended by Associate Editor R. Gupta.

A. B. Kahng is with the Computer Science and Engineering Deptartment, University of California at San Diego, La Jolla, CA 92093-0114 USA.

J. Lach is with the Department of Electrical and Computer Engineering, University of Virginia, Charlottesville, VA 22904 USA.

W. H. Mangione-Smith is with the Department of Electrical Engineering, University of California, Los Angeles, CA 90095-1594 USA.

S. Mantik, M. Potkonjak, and G. Wolfe are with the Department of Computer Science, University of California, Los Angeles, CA 90095-1596.

I. L. Markov is with the Electrical Engineering and Computer Science Department, University of Michigan, Ann Arbor, MI 48109-2122 USA.

P. Tucker was with the Computer Science and Engineering Deptartment, University of California at San Diego, La Jolla, CA 92093-0114 USA. He is now with Digial Integrity, Inc., San Mateo, CA 94403 USA

H. Wang is with Sun Microsystems, Inc., Palo Alto, CA 94303 USA.

Publisher Item Identifier S 0278-0070(01)08882-0.

I. Introduction

THE ADVANCE of processing technology has led to a rapid increase in integrated circuit (IC) design complexity. The economic drivers are compelling—only by putting more integration and more function on a single die and by achieving more revenue per wafer can multibillion dollar foundries be amortized over their useful lifespan. At the same time, market forces have led to more design starts, shorter design cycle times, and greater time-to-market pressures. Industry organizations have documented a compounding "design productivity shortfall" [47], which demands ever-larger design teams with each successive process generation just to maintain a given level of design competitiveness.[1] Finally, system design costs are increasingly impacted by software, which accounts for up to 70% of total development cost in recent design projects.

In response to these trends, *reuse-based* design methodologies for both hardware and software have been embraced as a means of achieving design productivity on par with the underlying silicon technology. The reuse-based vision is predicated on easily accessible, easily integrable "virtual components." Pure intellectual property (IP) companies, third-party application-specific integrated circuit (ASIC) libraries, tools for IP integration, and industry organizations such as the Virtual Socket Interface (VSI) Alliance[2] have created high expectations for the value and reusability of design IP. Nonetheless, a recognized obstacle to reuse-based methodologies is the lack of mechanisms to protect the rights of IP creators and owners.[3]

From both the research and implementation points of view, intellectual property protection (IPP) poses a unique set of new requirements that must be addressed by mathematically sound yet practical techniques. In this paper, we establish principles for development of new *watermarking-based* IPP procedures. A *design watermark* is an invisible (i.e., imperceptible to human or machine analysis) identification code that is permanently embedded as an integral part within a design. Desiderata for a given watermarking-based IPP technique, as determined by VSI Alliance—a leading industry organization—include [56]:

[1]According to [47], the available transistor density has increased by 58%/year over the last 20 years; designer efficiency (measured in transistors designed per staff-month) has increased by only 21%/year over the same period.

[2]http//:www.vsi.org

[3]For example, the VSI Alliance has identified six key technologies that must be in place to enable industrial-strength virtual component-based synthesis. In addition to system verification, mixed-signal design integration, on-chip bus, manufacturing-related test, and system-level design, IP protection is considered to be a crucial enabling technology [56].

0278–0070/01$10.00 © 2001 IEEE

1) maintenance of functional correctness;
2) transparency to existing design flows;
3) minimal overhead cost;
4) enforceability;
5) flexibility in providing a spectrum of protection levels;
6) persistence;
7) invisibility;
8) proportional component protection.

Our approach to IP watermarking applies to solutions of hard optimization and constraint-satisfaction design problems. It is centered around the use of *constraints* to "sign" the output of a given design synthesis or optimization. Namely, we say that solutions of a given optimization instance that satisfy these constraints *have a watermark embedded in them* and provide a probabilistic proof of authorship. The less likely that randomly chosen solutions are to satisfy these constraints, the stronger the proof of authorship is. A watermark's resistance to attacks is inversely proportional to an adversary's ability to manipulate it without resolving a given optimization problem from scratch. In practice, such approaches can be used to watermark particular very large scale integration (VLSI) designs as well as design tools, i.e., every design produced by a given tool can have the tool's watermark embedded in it. This approach is compatible with current IP development tools infrastructure and can be applied to protect both hardware IP and software IP, e.g., in Verilog and C++.

A. Motivating Example—3SAT

We illustrate key ideas behind watermarking-based IPP using satisfiability (SAT)—a classical NP-complete constraint-satisfaction problem [23] with numerous applications in VLSI design.

SAT (U, C):

Instance: A finite set of variables U and a collection $C = \{c_1, c_2, \ldots, c_m\}$ of clauses over U.

Question: Is there a truth assignment for U that satisfies all the clauses in C?

For example, $U = \{u_1, u_2\}$ and $C = \{\{u_1, u_2\}, \{\bar{u}_1\}, \{\bar{u}_1, \bar{u}_2\}\}$ is a SAT instance for which the answer is positive (a satisfying truth assignment is $t(u_1) = F$ and $t(u_2) = T$). On the other hand, if we have collection $C' = \{\{\bar{u}_1, u_2\}, \{\bar{u}_1, \bar{u}_2\}, \{u_1\}\}$, the answer is negative. SAT is well known as the first problem shown to be NP-complete and the starting point for establishing the known body of NP-completeness results [23]. Problems from many application domains have been modeled as SAT instances. In VLSI computer-aided design, SAT formulations have been used in testing [12], [21], [24], [37], logic synthesis, and physical design [21].

We now illustrate the *constraint-based watermarking* of a SAT solution. For convenience, we assume the 3SAT restriction of the problem, where each clause has exactly three variables. Consider the following 3SAT instance:

$$
\begin{aligned}
U =& \{u_1, u_2, \ldots, u_{14}\} \\
C =& \{\{\bar{u}_1 \bar{u}_2 u_9\}, \{\bar{u}_1 \bar{u}_3 \bar{u}_4\}, \{\bar{u}_1 u_2 \bar{u}_5\} \\
& \{u_1 \bar{u}_2 u_{10}\}, \{\bar{u}_1 \bar{u}_3 u_8\}, \{\bar{u}_1 \bar{u}_3 u_7\} \\
& \{u_1 \bar{u}_5 u_7\}, \{\bar{u}_1 \bar{u}_6 \bar{u}_{12}\}, \{\bar{u}_1 u_{10} u_{12}\} \\
& \{\bar{u}_1 u_6 u_9\}, \{\bar{u}_2 \bar{u}_3 \bar{u}_{10}\}, \{u_2 \bar{u}_5 \bar{u}_{14}\} \\
& \{\bar{u}_2 u_7 u_8\}, \{u_2 \bar{u}_8 u_9\}, \{u_3 u_4 u_8\} \\
& \{u_3 u_5 \bar{u}_7\}, \{\bar{u}_3 u_8 u_{13}\}, \{u_3 \bar{u}_9 \bar{u}_{11}\} \\
& \{u_3 u_{10} \bar{u}_{12}\}, \{\bar{u}_4 \bar{u}_7 \bar{u}_8\}, \{\bar{u}_5 \bar{u}_8 \bar{u}_{12}\} \\
& \{u_4 \bar{u}_7 u_{13}\}, \{\bar{u}_5 \bar{u}_9 \bar{u}_{11}\}, \{\bar{u}_5 u_7 u_9\} \\
& \{u_6 u_{10} u_{11}\}, \{u_6 \bar{u}_8 \bar{u}_{12}\}, \{u_7 u_9 \bar{u}_{12}\} \\
& \{u_7 u_9 \bar{u}_{13}\}, \{u_9 u_{11} \bar{u}_{14}\}, \{u_{10} u_{11} \bar{u}_{12}\}\}.
\end{aligned}
$$

Our goal is to alter the given 3SAT instance such that: 1) any satisfying assignment ("solution") to the modified instance is a solution to the original instance and 2) both the modified instance and the solution contain information (i.e., a "signature") that uniquely identifies the author of the solution.

Enumeration of the solution space indicates that the given 3SAT instance has 556 different satisfying assignments. We impose additional *constraints* in the form of extra three literal clauses, using the simple (case-insensitive) encoding $A - u_1, B - \bar{u}_1, C - u_2, D - \bar{u}_2, \ldots, Y - u_{13}, Z - \bar{u}_{13}$, space $- u_{14}$ to encode a signature. For example, the signature "cat dog fox" would be encoded using the extra clauses $\{\{u_2, u_1, \bar{u}_{10}\}, \{u_{14}, \bar{u}_2, u_8\}, \{u_4, u_{14}, \bar{u}_3\}, \{u_8, \bar{u}_{12}, u_{14}\}\}$. Here, the end of the message is padded with an extra space to maintain three literals per clause.

The signature "Watermarking Techniques for Intellectual Property Protection University of California at Los Angeles VLSI CAD LAB" adds 38 new clauses to the instance and decreases the number of satisfying assignments from 556 to two. We claim that any satisfying assignment for this augmented 3SAT instance contains our signature and that the likelihood of someone else generating such a solution by chance is two in 556, or 0.00496. In this example, the addition of a watermark incurs no overhead; it simply guides which solution is selected. Obviously, watermarked solutions exist only for watermarks of small enough size. The larger a given instance, the larger watermarks can be embedded into it. We note that our watermarking strategy is based on *preprocessing* of the input instance and is *nonintrusive* in that any existing solution method remains applicable to the augmented (watermarked) 3SAT instance.

In particular, any SAT solvers from the four major classes can be used: 1) randomized local search [46]; 2) exact deterministic methods based on resolution [19] and branch-and-bound; 3) nonlinear programming relaxation and rounding [25]; and 4) a variety of binary-decision-diagram-based techniques [11]. In our experience, many commonly encountered NP-complete formulations can also be watermarked using similar constraints.

B. General Approach

The above example can be extended to generic optimization and constraint-satisfaction problems by viewing watermarks as ways to limit the set of possible solutions to a small subset. The smaller the probability of selecting a "watermarked" solution by chance, the stronger the watermark. While making the "watermarked subset" smaller would generally improve the strength of the watermark, one must ensure that this set is nonempty. As

far as the objective function or functions are concerned, the watermarked solutions must not be inferior to average solutions; otherwise, such a watermark will be too costly to use.

Our paper proposes to embed watermarks by adding constraints to optimization and constraint-satisfaction problems. An outline of a generic watermarking procedure is given in Section III along with principles of constructing specific IPP protocols and various considerations that arise in practice. In particular, watermarks must withstand a number of attacks, described in Section III-E.

C. Applications—Domain-Specific IPP Protocols

Using the concept of constraint-based watermarking, we develop new protocols for IPP in the domains of *system-level* and *physical design*. These domains areas were chosen because they are both natural for watermarking applications and challenging as optimization problems.

1) System-level and physical design are traditionally viewed as "difficult" domains. Mathematically, many design problems in these domains contain NP-complete problems (SAT, graph coloring, vertex ordering, routing, etc) and, in practice, even a small percentage variation in solution quality can make or break a design. The sheer difficulty of finding good solutions increases the cost of those solutions, thus there is more interest in protecting them.
2) High-quality solutions in system-level and physical design often have strong structural resemblance to each other [15]. Therefore, it is challenging to devise a watermarking technique that can dramatically decrease the number of solutions without compromising solution quality.
3) With deep-submicrometer technology, many performance constraints (e.g., budgeted edge delays consistent with path timing bounds) cannot be considered satisfied until they are satisfied in the physical design. Thus, for example, it may be insufficient to "watermark" a design by constraining timing budgets without verifying that such constraints are satisfied after physical design.
4) Other trends—IP reuse methodologies, higher perceived valuation of "hard IP," increasing availability of multiple foundry sources, difficulty of performance validation before physical design, changing handoff models, etc.—all point to physical design as an appropriate juncture in the design cycle for watermarking.

The empirical evaluation of the proposed techniques is performed using placement and routing applications. For placement, we propose a *postprocessing* flow that encodes a signature as specified parity (i.e., odd- or even-index) of the cell row within which particular standard cells must be placed. For routing, we propose a *preprocessing* flow that encodes a signature as upper bounds on the wrong-way wiring used to route particular signal nets. Using real industrial design examples and commercial layout tools, we demonstrate the effectiveness of both the preprocessing- and the postprocessing-based watermarking. In particular, strong signatures are achieved without compromising any of the standard metrics for solution quality [routability, wirelength, number of vias, central processing unit (CPU) time, etc.]. For placement watermarking, we demonstrate that adding signatures has no negative effect on timing quality for a timing-driven test case. We also demonstrate that both our placement and routing watermarking techniques are tamper-resistant. We conclude that addressing IP protection at a lower level of abstraction has an advantage: designs inherently have orders of magnitude more components, allowing significantly stronger proofs of authorship as well as lower overhead. We also conclude that the postprocessing approach is not only feasible, but indeed quite attractive for several reasons: 1) it enables watermarking of already existing designs; 2) it enables direct calculation of the hardware overhead incurred by IPP; and 3) it may be likelier to find acceptance among designers and managers, since the complete design process is not altered in any way.

D. Organization of the Paper

The remainder of this paper is organized as follows. Related concepts in artifact watermarking, cryptography, and IP-based synthesis are surveyed in Section II. Principles and desiderata (e.g., protection requirements) of nonintrusive constraint-based IP watermarking are discussed in Section III. This section also introduces probabilistic proofs of authorship for watermarks and classifies typical attacks. Section IV illustrates wide-ranging applicability of the proposed watermarking techniques and offers an in-depth discussion of watermarking in the VLSI physical design domain. This discussion is continued in Section V, where a physical design flow with watermarking is given. Experimental results, including the actual strength of watermarks and resistance to tampering, are given in Sections V and VI. Finally, we conclude in Section VII that constraint-based watermarking has significant potential to protect IP and support design reuse.

II. Related Work

Related work can be summarized with respect to four research and development literatures: 1) watermarking techniques for IPP; (2) cryptographical techniques and tools; (3) IP-based synthesis; and (4) VLSI physical design techniques. Recent work of Charbon and Torunoglu on watermarking-based IPP in VLSI physical design is contrasted with our present work in Sections IV-D–F.

A. Watermarking

Recently, a number of techniques have been proposed for data hiding in image, video, text, and audio data. For example, data hiding has been proposed as a mechanism for embedding important control, service, or reference information in particular data. However, there is a wide consensus that IPP is the prime application of watermarking. A *watermark* is a mechanism for embedding additional information into an artifact (text, image, video, audio) or piece of IP (hardware, software, algorithm, data organization) that is: 1) designed to identify the author, the source, the used tools, and techniques and/or recipient of the artifact or the IP and 2) difficult to detect and remove. More than 50 different watermarking techniques for protection of images have been proposed [7], [10], [16], [20], [26], [50],

[54]. While the majority of these exploit imperfections of the human visual system to embed invisible watermarks, several works address advantages of visible watermarks [9]. Although many of the initial image watermarking techniques were not robust enough to provide proper proof of ownership [18], several recent image watermarking techniques are quite strong in this regard [53]. Several data hiding techniques have been proposed that exploit frequency and time imperfection of the human auditory system [3], [6], [16]. AT&T researchers have developed a number of techniques for watermarking of text documents [4], [9]. Video-on-demand research has resulted in a suite of approaches for watermarking video, mainly MPEG-2, streams [27], [28], [48].

All of the cited references treat only watermarking of static data that is eventually consumed by a human. It is important to distinguish traditional requirements for *artifact watermarking* from those governing the *IP protection* applications that we address. Artifact watermarking simply adds a signature into a given artifact *without regard to maintaining correctness or function*. "Transparency" of the signature stems from imperfections in human auditory and visual systems: the artifact (e.g., a digitized photograph) is actually changed, but the human eye cannot perceive the change. While artifact watermarking has been used for thousands of years, only with the proliferation of digital media has it attracted wide research and economic interest, e.g., for protection of audio [3], [31], text [8], [38], image [17], and video.

In contrast, watermarking for IP protection imposes much stronger constraints because the watermarked IP must remain *functionally correct*. For example, one cannot arbitrarily introduce extra lines of code into a Verilog program or extra devices and interconnects into a transistor-level layout. Our discussion is centered around the following key idea: watermarking for IPP is most practically accomplished by imposing a set of additional *constraints* during the design and implementation of IP, so as to uniquely encode the signature of the author. Since 1996, the effectiveness of this generic scheme for watermarking-based IPP has been demonstrated at the level of algorithms [31], behavior [30], logic synthesis, and physical design, as well as in field-programmable gate array (FPGA) designs [35], [36].

B. Cryptography

Public-key techniques, which exploit computationally intractable problems as a basic building block, revolutionized cryptography by introducing convenient tools for secure communications over insecure channels. The idea was introduced by Diffie and Hellman [22]. Soon after that, Rivest *et al.* developed the number factoring-based suite of techniques which, somewhat modified, successfully passed numerous attacks to become the *de facto* standard for modern cryptographic techniques. Since 1976, cryptographic algorithms and techniques have evolved through vigorous innovation and public scrutiny, resulting in a variety of digital signature mechanisms, as well as protocols for secret splitting, timestamping, proxy signatures, group signatures, key escrow, oblivious transfer, oblivious signatures, digital cash, etc. [39], [49].

Several cryptographic techniques are directly relevant to our design watermarking approach. Cryptography provides the theoretical foundations as well as the algorithmic and protocol infrastructure that support watermarking-based IPP and provide a wide spectrum of authorship protection services. The present paper uses cryptography tools for generating a set of physical design constraints that correspond to the signature of the author of the design. Our use of cryptographic techniques ensures cryptographically strong hiding and decorrelation of the added signature constraints. Specifically, we use for these two tasks the cryptographic hash function MD5, the public-key cryptosystem RSA, and the stream cipher RC4 [39], [45] on which many of today's state-of-the-art cryptographic commercial programs are based [45].

C. IP-Based Synthesis

As noted above, short design times, increased device counts and design starts, and foundry amortization have together forced a change in design methodology. The new semiconductor business regime is based on IP reuse. No other regime is compatible with rapid turnaround and high device counts; no other regime enables ASIC vendors to keep their foundries full of high-value product.

Less than two years ago, the VSI Alliance and CFI Component Information Library Project were first announced. Today, at least three major industry organizations—RAPID (IP providers),[4] SI^2 (ASIC vendors),[5] and VSI Alliance [a large organization of electronic design automation (EDA) vendors, ASIC vendors, system houses, and IP providers]—are actively building the industry infrastructure for IP-based design.[6] Several missing infrastructure pieces are technical, with deep implications for the associated EDA technology and design methodologies.[7] Other missing pieces include the standards for representing design IP. However, arguably the most pressing infrastructure issues are legal: what are the risks faced by ASIC suppliers and EDA tools vendors as they incorporate third-party IP? Who holds accountability for design success? How will the rights of IP creators and owners be protected? It is notable that despite their varying perspectives, each of the three major industry organizations has a working group for legal issues.

D. Related Physical Design Techniques

Constraint specification and management now receive close attention through all phases of chip implementation, including physical design. This is at least partly due to the increasing effect of device and interconnect embedding on system perfor-

[4]http://www.rapid.org

[5]http://www.si2.org

[6]The early CFI effort spawned the Pinnacles Component Information Standard and CFI subsequently became SI^2 (Silicon Integration Initiative).

[7]For example, how reusable IP will be bundled with standardized test and simulation "envelopes" or the form of reusable IP and the manner in which it will "mix and match" remains unclear. Current visions encompass varying degrees of "hardness" of the IP, e.g., soft (HDL program), medium (HDL program + floorplan), hard (GDSII stream file), etc. Harder forms of IP might have greater value since they would embody greater amounts of design effort. At the same time, hard IP is less reusable due to its well-defined shape and inherent timing/noise/thermal context; it also allows less flexibility in floorplanning and routing due to constraints on over-the-block routing (e.g., timing and signal integrity margins). It remains to be seen how "parameterizable" an IP block can be in terms of area-time tradeoffs, migration to alternate processes, routing resource utilization, etc.

mance (i.e., the "deep-submicrometer crisis"). Standard chip implementation flows begin with such high-level constraints as clock cycle times and offsets, input–output (I/O) boundary timing, power dissipation bounds, and choice of packaging and implementation technology. Derived constraints[8] will then arise throughout the register transfer level (RTL) floorplanning, block placement, and routing phases. Within physical design, timing[9] and physical (floorplanning)[10] constraints are most common.

The mechanisms by which physical design tools enforce such constraints vary widely. However, classic paradigms such as top-down min-cut placement synthesis or ripup-and-reroute interconnect synthesis generally do not support constraints well. One reason is that iterative search mechanisms used today were originally adopted for regimes with "smooth" cost surfaces, while adding constraints induces more zero–one "discontinuities" in the cost surface. Another reason is that many performance constraints are "global" (e.g., path-delay, power dissipation, electromagnetic coupling, signal integrity) while current layout approaches rely on local optimizations. Most importantly, "good" solutions to hard combinatorial problems are often quite similar.[11] The implications for watermarking in physical design are that: 1) current tools do not easily support too many "extra" watermarking constraints and 2) introduction of too many watermarking constraints will likely degrade solution quality. These issues complicate the choice of watermarking technique.

III. Precepts and a General Approach to Constraint-Based IPP

In this section, we develop basic precepts and a general constraint-based approach for watermarking IP protection. Our discussion will abstract the design process as a form of optimization and we will focus on opportunities for nonintrusive watermarking (i.e., methods that can be transparently integrated within existing design flows via preprocessing or postprocessing).

A. Context for Watermarking

The following ingredients form the *context* for a nonintrusive watermarking procedure.

1) An *optimization problem* with known difficult complexity, corresponding to some design synthesis task. By difficult, we mean that either achieving an acceptable solution or enumerating enough acceptable solutions is prohibitively costly. The solution space of the optimization problem should be large enough to accommodate a digital watermark.
2) A well-defined *interpretation* of the solutions of the optimization problem as IP.
3) Existing *algorithms* and/or *off-the-shelf software* that solve the *optimization problem*, likely without any kind of watermarking involved. Typically, the "black box" software model is appropriate and is moreover compatible with defining the watermarking procedure by composition with preprocessing and postprocessing stages.[12]
4) *Protection requirements* that are largely similar to well-understood protection requirements for currency watermarking. As discussed below, such requirements include: a) removing or forging a watermark must be as hard as recreating the design; and b) tampering with a watermark must be provable in court.

A nonintrusive watermarking procedure then applies to any given instance of the optimization problem and can be attached to any specific algorithms and/or software solving it. Such a procedure can be described by the following components.

1) A *use model* or *protocols* for the watermarking procedure. This is not the same as algorithm descriptions; it is less formal and can be helpful in revealing possible attacks beyond the generic types noted above. For example, algorithms assume a cell numbering, while renumbering cells can defeat a watermarking procedure (something that can be seen only at the protocol level). In general, each watermarking scheme must be aware of attacks based on design symmetries, renaming, reordering, small perturbations (which may set requirements for the structure of the solution space), etc.
2) Algorithmic descriptions of the *preprocessing* and *postprocessing* steps of the watermarking procedure.
3) *Strength and feasibility analyses* showing that the procedure satisfies given protection requirements on a given instance. Strength analysis requires metrics, as well as structural understanding of the solution space [e.g., "barriers" (with respect to local search) between acceptable solutions]. Feasibility analysis requires measures of solution quality, whether a watermarked solution remains well formed, etc.
4) *General robustness analyzes*, including discussion of susceptibility to typical attacks, discussion of possible new

[8]Derived constraints are of two basic types. *Inferred* constraints can often be viewed as "transformed," e.g., when a signal net's wirelength upper bound is inferred from a signal propagation delay upper bound. *Refined* constraints can often be viewed as created by a "budgeting" or "allocation" process, e.g., when a global path-delay constraint is broken up into separate edge delay constraints.

[9]Path-delay constraints are often expressed in some form of standard delay format (SDF) with heuristic "path cover" techniques used to reduce data volume and improve convergence of timing-driven layout tools. A static timing analysis engine may operate directly from the clock cycle times/offsets and I/O boundary timing to evaluate timing correctness, without explicit enumeration of timing path constraints. For purposes of layout design, path-delay constraints are typically budgeted into individual constraints on source-sink edges [52].

[10]To improve timing convergence of the design process, assumptions made during RTL floorplanning or block floorplanning must be propagated to downstream flow stages (e.g., placement and global routing). This is often accomplished via region constraints: a given cell must be located in a given region of the layout, a set of cells must be colocated as a "group," etc. Such constraints may be captured using physical design exchange format (PDEF) or equivalent formats which allow specification of assumed routing topology, layer usage, etc. at the level of global routing.

[11]This has been generally characterized as a "big valley" [5] or "massif central" [33]; the phenomenon has also been specifically documented for standard-cell placements under the minimum wirelength objective [15].

[12]Watermarking the results of nondeterministic and/or unknown algorithms—or even "handmade" results—is possible as well. IP protection can even be achieved, to some extent, with black-box off-the-shelf software that is viewed as a one-way function mapping inputs to design solutions. In this discussion, we focus only on the simple model involving known deterministic algorithms.

attacks, performance guarantees (including complexity analysis), and implementation feasibility.

Before describing a general strategy for embedding digital watermarks, we observe that optimization problems with known watermarking procedures share several common features: 1) having multiple acceptable solutions (we typically accept suboptimal solutions for NP-hard problems); 2) solved by optimization heuristics; and 3) discrete in nature. While "continuous watermarking" is possible as well, e.g., by mapping into "discrete watermarking" by Fourier transform, it is beyond the scope of our work.

B. General Strategy for Constraint-Based IPP

Our general strategy is to map an author's signature into a set of constraints ("desired relations") that can independently hold for a particular solution (or independence can be assumed, via some manipulations). If disproportionately many of these constraints are satisfied, the presence of the signature is indicated and vice versa. Choosing the type of constraints and the tactic (e.g., preprocessing or postprocessing) by which we make it likely for more of them to be satisfied than would otherwise be expected is what instantiates a particular watermarking algorithm from the general strategy. These choices can dramatically affect the strength of the watermark and the degradation of solution quality caused by watermarking. To facilitate later discussion, we now describe generic watermarking and signature verification procedures using "Alice (and Bob)" scenarios, where Alice uses watermarking to protect some IP (below, Bob will attempt to subvert such protection).

1) Generic Watermarking Procedure: Alice wishes to protect some IP that involves many stages of processing. She chooses to watermark one or more of these stages. The results of these stages now carry a watermark which will propagate down to the output of further stages all the way down to the final result. Clearly, the amount of watermarking she imposes on a particular stage trades off with the degree of degradation of quality of the final result. Alice watermarks each stage by selecting a set of "constraints," then uses preprocessing of the stage's input and postprocessing of the stage's output to encourage a disproportionate number of these constraints to be satisfied. Note that Alice need not tell anyone which constraints correspond to her signature. In addition, numerous IPP protocols can be built on top of the basic watermarking scheme. For example, one can encode ID of both licensee and licenser and therefore provide mutual protection of both sides in the transaction.

2) Generic Signature Verification Procedure: To demonstrate that a particular stage was watermarked Alice must show that its solution (which may have been passed on undisturbed to other stages and perhaps all the way to the final result) satisfies a disproportionate number of her watermarking constraints. By identifying the watermarking constraints, determining how many of them are satisfied and calculating P_c—the probability of so many (or more) of the constraints being satisfied by coincidence—Alice can verify that her signature is present. A strong proof of authorship corresponds to a low value for P_c. Note that to show this to other people, Alice must reveal her signature and, hence, the chosen constraints.

C. Selection of Constraints

Selection of constraints is a critical step in our proposed watermarking approach. In this step, a given signature is mapped into a set of constraints and the choice of a particular mapping is responsible for the strength of the watermark in a watermarked solution.

We propose constructing such mappings using pseudorandom number generators that allow selecting a set of X constraints independently (for different parts of signature). It is only slightly more work to select a set of X constraints with no constraint repeated and pseudorandomly otherwise. Thus, the task of mapping an author's signature into a set of constraints can be reduced to the task of seeding a pseudorandom number generator with the signature. For the latter, suppose that the author's signature is a particular text message. We can convert this message into a cryptographically sound pseudorandom bit stream by simply hashing the message with a cryptographic one-way hash function, such as MD5 [44] and using the hash as a seed for a stream cipher, such as RC4.[13]

D. Analysis—Proof of Authorship

A watermark's *proof of authorship* is expressed as a single value P_c, which is the probability of so many (or more) of the selected constraints being satisfied. Essentially, P_c is the probability of a nonwatermarked solution carrying our watermark by coincidence. We wish this probability to be convincingly low so as to have a strong proof of authorship. When we cannot compute P_c exactly, it is acceptable to overestimate it so that we actually report an upper bound on P_c. Computing such an upper bound on P_c is typically straightforward. Let p be the probability of satisfying a single random constraint by coincidence. This value or a fairly tight upper bound on it is usually obvious from the definition of a constraint. Here, we assume that p is independent of whether the other constraints were satisfied. Let C be the number of imposed constraints. Let b be the number of these constraints that were *not* satisfied. Let X be a random variable that represents how many of the C constraints were not satisfied. Now, P_c can be computed as a sum of binomials, i.e., the probability that coincidentally only b or less of C constraints were not satisfied is given by $P_c \equiv P(X \leq b) = \sum_{i=0}^{b}((C!/(C-i)! \cdot i!) \cdot (p)^{C-i} \cdot (1-p)^i)$. This analysis assumes that p is independent of whether other constraints are satisfied, an assumption that is often untrue. However, when the number of imposed constraints (C) is sufficiently small, we have a very good approximation. For detailed explanation of this calculation, see [42] and [43].

E. Analysis—Typical Attacks

There are several general ways of attacking our watermarking scheme. Here, we discuss the more prominent ones: finding "ghost signatures," tampering, and forging. We analyze these attacks using "Alice and Bob" scenarios.

1) Attack—Finding Ghosts: Bob wishes to steal IP from Alice and claim it as his own. He knows that Alice has protected her IP (i.e., the solution to a particular stage of the design process) with a watermark, but will claim that the IP

[13]http://www.scramdisk.clara.net/d_crypto.html

also contains his own watermark. Bob, thus, attempts to find a *ghost signature*, namely, a signature that corresponds to a set of constraints that yields a favorable P_c, but which was discovered after fact instead of being actually watermarked into the solution. To be convincing, Bob must find a ghost signature that yields a sufficiently convincing value P_c.

Bob has only two approaches. He may choose a set of constraints (presumably ones that yield a good proof of authorship P_c) and then attempt to find a signature that corresponds to this set. This requires reversing the cryptographically secure one-way functions that convert a signature into a set of constraints, which is hard. All signatures before embedding are processed using a one-way function. Therefore, finding a ghost signature is at least as difficult as breaking a selected cryptographical protocol, which is widely considered to be a practically impossible task. Alternatively, Bob may try a brute-force approach to find a signature that corresponds to a set of constraints that yields a convincing proof of authorship P_c. However, this brute-force attack becomes computationally infeasible if the threshold for proof of authorship is set sufficiently low (e.g., $P_c \leq 2^{-56}$).

2) Attack—Tampering: If Bob cannot find a convincing ghost signature, he may decide to *tamper* with Alice's solution. Ideally, such tampering would completely remove Alice's signature and add Bob's own signature. Bob can do this by simply resolving the problem from scratch with his own watermark encoded then continue through subsequent processing stages based upon the output he obtains. Nothing can be done to stop this directly. However, we believe that in realistic scenarios, Bob cannot afford to redo all of the subsequent phases of the design process, particularly if the watermarking occurred relatively early in the process.

There are realistic means by which Bob can tamper with a solution without having to resolve every subsequent stage of the process. Generally, these amount to transforming the solution output by the last phase of the design process, where the transformation has a similar effect on the output of the watermarked phase of the design process. Specific changes that Bob makes to the final solution will likely correspond to: 1) local perturbations of the solution to the watermarked phase or 2) global-scale transformations such as those that exploit a symmetry of the design representation. Given that Bob is limited to these kinds of tampering attacks, it is critical that Alice's watermarking technique be resistant to such transformations. Note that since the attacker does not know which constraints correspond to the author's signature, tampering attacks might not be able to ruin the proof of authorship before they significantly degrade the quality of the final solution, rendering tampered solution useless.However, by tampering, an attacker may be able to: 1) remove a signature that is known to him or 2) add an entirely new signature.

3) Attack—Forging : Finally, Bob may attempt to subvert Alice's watermark by inappropriately watermarking other solutions with Alice's watermark. In other words, Bob wishes to *forge* Alice's signature. To do this, Bob needs a signature that he can convince others belongs to Alice. If a signature corresponds simply to a text message (as it has so far in this discussion) then Bob's task is easy: he simply chooses a text message resembling one that Alice would use. However, such attacks can be easily prevented by using a public key encryption system [41]. Any message actually signed by Alice would be encrypted with her private key, yet verifiable with her public key. Notice that the private key is not compromised even if messages that are encoded with it are compromised, so Alice may still demonstrate the presence of her watermark to anyone who knows her public key without compromising her private key. Thus, Bob is able to forge a message from Alice only if he knows her private key. Note that the verification procedure does not require decryption of potentially altered design. All that is needed is to compare the level of similarity in terms of the number of satisfied constraints. If a large percentage of (not necessarily all) constraints are satisfied, this provides proof of similarity. Decryption is done on the unaltered design to establish the connection between the solution and the author.

IV. IP Watermarking Examples

The wide-ranging applicability of the principles developed above is illustrated in this section on three examples of IP watermarking in unrelated areas—system-level design, FPGA design, and the management of path-timing constraints. Those examples are followed by an in-depth discussion of watermarking techniques in standard-cell place and route.

A. Preprocessing-Based Watermarking Applied to System-Level Design Steps

At the system level, instruction and data caches consume a significant portion of the overall area and often have crucial impact on system timing and power consumption [34]. Much effort has been devoted to allocating minimal cache structures and optimizing code for effective cache utilization [57]. A particularly successful technique is *cache-line coloring* [29].

Given a code segment and input data benchmarks, cache-line-coloring code optimization seeks a permutation of basic block code segments such that the mapping of code to cache entries minimizes the cache-miss ratio over the given benchmarks. The problem can be modeled as follows. The program is profiled with respect to the benchmark data and spatial (frequent sequences of sequentially executed code) and temporal (frequent control sequences) correlations noted among basic blocks of code. The program is modeled using a control data-flow graph, where a graph node corresponds to a set of instructions that are encompassed in a single basic block and fit exactly one cache line. Weighted edges between nodes correspond to spatial or temporal correlations that exceed given threshold values (modeling accuracy, thus, depends on the thresholds for edge inclusion). The problem of minimizing cache misses is equivalent to finding a solution to graph coloring using a given fixed number of colors (corresponding to available cache lines). This optimization can result in significant performance increase, as experimentally shown by Kirovski *et al.* [34] and can play an important role in the design of modern multimedia, communications, or low-power systems-on-silicon.

To watermark such designs, the initial design constraints may be augmented with additional constraints corresponding to the digital signature of the designer. For example, following the

technique for watermarking of graph coloring solutions proposed by Hong and Potkonjak [30], one may add additional edges to the graph according to some encrypted signature of the author. Therefore, the signature will be embedded in the activation path which transfers data between two levels of hierarchy.

Graph (or some other object) isomorphism is a technique that can and has been used to establish similarity between two artifacts, e.g., software programs [32], [40]. The key difference between this type of technique and watermarking is that plagiarism detection techniques do not provide any indication of which program is a copy of which: they only point out similarity. In addition, these techniques require solution of intractable combinatorial problems. While plagiarism detection techniques have their role and can be applied in conjunction with watermarking techniques, their effectiveness is subsumed by watermarking-based IPP techniques.

It is important to note that watermarking protection against attacks is greatly enhanced by the layered nature of design process: each change at a higher stage of the design process implies the need for great investment in redoing all later steps. In light of this fact, it is much more important to focus on metrics for proof establishment than on metrics for attack resiliency.

B. Postprocessing in Physical-Level FPGA Design

One method of watermarking an FPGA at the physical level involves manipulating unused portions of the configuration bitstream. Informed parties can then extract the mark from the bitstream. There is no effect on the function of the design during insertion or extraction because only unused portions of the design are altered. This approach can be implemented through preprocessing, iterative, or post processing techniques. The advantage of postprocessing is that it does not impact other computer-aided design tools and has zero impact on design performance, area or power consumption. The disadvantage of this approach is that the watermark is not embedded in the functional part of the design; given enough information, the watermark can be removed without affecting design functionality. An example of an iterative approach can be found in the work by Lach *et al.* [35], [36].

An example of a purely postprocessing approach involves inserting the watermark into the control bits for unused outputs from configurable logic blocks (CLBs). Certain bits in the configuration bitstream that control multiplexers for the CLB outputs can be replaced by watermark bits if the CLB outputs are not used. For example, the Xilinx 4000 family of FPGAs contain CLBs with four outputs [58]. Two outputs (X and Y) are combinational, while the others (XQ and YQ) can be used in sequential designs. The two combinatorial outputs are each controlled by a two-to-one multiplexer and the two sequential outputs are each controlled by three two-to-one multiplexers and one four-to-one multiplexer. Fig. 1 shows the control layout of the 4000 family's CLB outputs.

The number of configuration bits associated with a multiplexer is equal to (or greater than) the number of required control bits. Therefore, one and two watermark bits can be inserted at each unused two-to-one and four-to-one multiplexer, respectively. Thus, each unused combinatorial output can store one watermark bit and each unused sequential output can store five

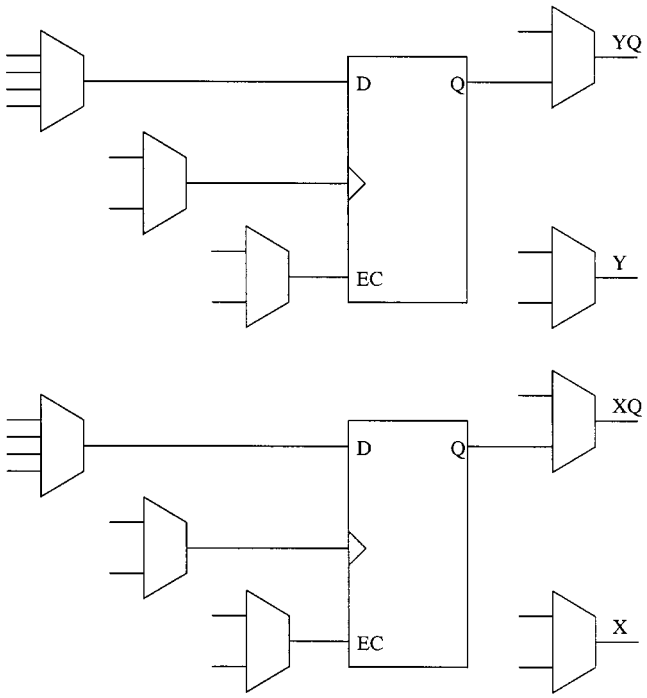


Fig. 1. Control directly attributed to CLB outputs.

TABLE I
NUMBER OF BITS AVAILABLE FOR WATERMARKING

% Outputs Unused	Part/# CLBs 4006 /256	4010 /400	4013 /576	4020 /784	4025 /1024
1	30	48	69	94	122
5	153	240	345	470	614
10	307	480	691	940	1228
20	614	960	1382	1881	2457

watermark bits. The total number of watermark bits that can be inserted in an entirely unused CLB is 12. Table I shows the number of watermark bits that can be inserted into various devices within the 4000 family given certain percentages of unused CLB outputs. The numbers calculated here are for an even number of unused combinatorial and sequential outputs.

The process of watermark insertion in this approach is an entirely postprocessing step and requires very little added design effort. The tool methodically scans the bitstream, searching for unused outputs by finding CLB output pinwires that do not attach to any external CLB interconnect. Upon the detection of unused outputs, the next bits of the watermark are inserted in place of the corresponding multiplexer configuration bits. The size of the watermark is limited by the number of bits made available by this approach. Extracting the watermark is an almost identical process. The tool finds unused CLB outputs the same way as was done in insertion and pieces the watermark back together by examining the corresponding multiplexer configuration bits.

This FPGA watermarking approach requires little extra design effort, can store fairly large watermarks, allows for easy mark extraction, and has no overhead in terms of design area

or performance. However, because a mark is nonfunctional, it may be removed by reverse engineering a design to a stage in the flow before the mark has been applied. Fortunately, most FPGA vendors will not reveal the specification of their configuration streams, specifically to complicate the task of reverse engineering and, thus, protect the investment of their customers. For example, the Xilinx XC4000 devices follow a form of Pareto's rule: the first 80% of the configuration information can be determined relatively easily by inspection, the next 18% is much more difficult, etc. The complexity is enhanced by an irregular pattern that is not consistent between rows or columns as a result of the hierarchical interconnect network. Xilinx does not take any specific actions to make their configurations difficult to reverse engineer. However, they do believe that it is difficult to do in general and they promise their customers that they will keep the bitstream specification confidential in order to raise the bar for reverse engineering [55].

C. Preprocessing of Path-Timing Constraints

Finally, in the context of physical design, we present a new preprocessing-based approach for design watermarking. Our approach exploits the flexibility with which *path-based timing constraints* can be satisfied.

Consider the typical elements of an input instance for timing-driven placement and routing:

1) physical floorplan, library of physical cell masters, and cell-level netlist;
2) cell-level performance macromodels for each cell master [e.g., nonlinear table models (Synopsys. lib, Cadence. ctlf, OVI ALF, etc.)] for timing and power dissipation analysis;
3) technology file (models of interconnect RC characteristics, design rules, etc.);
4) constraints, which are chiefly: a) "direct" placement and routing constraints (e.g., region-based location constraints arising from the floorplanner and transmitted in PDEF format) and b) performance constraints (e.g., SDF latch-to-latch path timing upper and lower bounds, with false path and multicycle constraints specially annotated).

We watermark a design by selecting path timing constraints and replacing them with "subpath" timing constraints. Suppose that we have the path timing constraint $t(C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow \cdots \rightarrow C_{10}) \leq 50$ ns ($C_i \equiv$ cells). We can allocate the timing bound between two subpaths and replace this constraint by two constraints $t(C_1 \rightarrow \cdots \rightarrow C_5) \leq 20$ ns and $t(C_5 \rightarrow \cdots \rightarrow C_{10}) \leq 30$ ns. All else being equal, the chance that satisfying the original constraint happens to satisfy both of these subpath constraints is at most one-half.[14] Constraining on the order of hundreds of timing paths (from the several millions one finds in typical verbose SDF specifications) is transparent to timing-driven design tools, yet affords strong proofs of authorship. Similar techniques can be applied in the regime of compact SDF timing constraints or at the budgeting stages of timing-driven design.[15]

D. IP Protection for Standard-Cell Place and Route

In this and the following two sections, we propose and give comprehensive validation of new mechanisms by which standard-cell physical design can be constrained. Our goal has been to develop a watermarking protocol that, beyond satisfying criteria listed above, is: 1) consistent with existing design practices and tools; 2) relatively easy to implement; and 3) acceptable in terms of its impact on real-world layout metrics.

We note that other authors have recently also addressed watermarking of physical design solutions. In particular, shortly after we submitted the conference version of this paper and presented our approach to watermarking, several works were submitted by Charbon. Reference [13] presents a formalization of the watermarking problem and algorithms for watermark generation and detection at several abstraction levels of the physical design process. The work of [13] also discusses the concepts of robustness against forgery and analyzed the proposed algorithms with respect to their robustness. Even more recently, Charbon and Torunoglu [14] enhanced this work to propose a method to reconstruct the original watermark for a given design.

As will be clear from what follows, there are a number of differences between our present work and that of Charbon. First, we address physical design for standard cells, while he addresses macro block-based design. Second, we embed our watermark during both placement and routing, while Charbon address only placement-related watermarking. Third, we propose both preprocessing and postprocessing watermarking techniques, while Charbon discusses only preprocessing schemes. Finally, we provide the first quantitative discussion of potential attack.

E. Row-Based Placement

For row-based placement, traditional physical embedding constraint types (i.e., region and grouping constraints) are straightforward to realize. Region constraints are transparent to top-down placers, since iterative partitioners accommodate "fixed" preassignments (see [2] for a review). Annealing placers (see [51] for a review) also support such constraints by restricting move generation and analytic placers support region constraints via inequalities or center-of-gravity constraints (see [1] for a review). Grouping constraints are typically enforced by inducing contracted netlists over clustered representations of the design. However, region and grouping constraints are not well suited to placement-based watermarking: when made

[14]Note that the allocation would be done with respect to available slack on the path, e.g., path-delay upper bound minus sum of "intrinsic" cell delays. Also note that constraint satisfaction will likely be determined in the context of final layout.

[15]In general, the physical design context presents a rich environment for constraint-based watermarking. For example, the physical library information and/or design rules allow variant pin access models for a cell, which will constrain how interconnects attach to pins; extra blockage geometries in cell instances or masters can also be used to constrain the routing and via and stub rules can again encode a signature within the output of a constraint-driven router. Simple parity-based schemes abound, e.g., based on mirroring of cells, parity of row indexes to which cells are assigned, routing of wires to the left or right of shield wires, etc. Even performance macromodels (nonlinear table models for timing and power) can be perturbed (thus, constraining the performance-driven layout) to influence the layout tool's output.

without any structural knowledge of the netlist or of good placement solutions, they can lead to substantial deterioration of solution quality. This deterioration can worsen when the design is performance-constrained, e.g., one cell arbitrarily chosen from timing-critical paths may be constrained in a "wrong" region of the layout, thus leading to violated timing-constraints.

Our approach bases the watermarking constraints on the underlying fine-grain placement substrate, which is well defined prior to the placement phase of design. By "placement substrate," we mean the row structure of legal site locations in the physical floorplan. In particular, we constrain individual cells to be placed with specified *cell row parity*. For example, cell INV4_10 might be constrained to be placed in a cell row that has EVEN index; cell RKPPX1Y might be constrained to an ODD-index row. Our approach has the following advantages.

1) Very few constraints are needed to make a strong signature. For example, if the signature constrains 50 cells with specific row parities and if the placer realizes all 50 constraints, the chances are 2^{-50} that this could have occurred by accident. Typical placement instances have tens of thousands of standard cells.
2) It is compatible with region and group constraint types and can be applied as soon as a gate-level netlist exists (no specific row/site plan is needed). *A priori*, the only time failure is guaranteed is when a "watermark cell" is constrained to be in, e.g., an EVEN row and is simultaneously constrained to be in a region that contains only a single ODD row.
3) It is not easy to tamper with the signature via local perturbations: the small signature size implies that many cells must be perturbed before the signature becomes unrecoverable. Furthermore, local perturbations that shift cells between cell rows are difficult to make without worsening the solution quality.
4) It is not affected by downstream stages of the design flow. Many current design methodologies do not significantly change the row assignments (let alone the locations) of existing cells during routing; hence, our proposed watermarking scheme will remain intact.
5) It allows the watermark to be realized completely during the placement phase. (For schemes such as the watermarking of budgeted timing constraints, the realization remains incomplete until after routing.)

F. Routing

For standard-cell routing, applicable constraints usually involve performance (e.g., crosstalk and delay bounds) or reliability (antenna rules, electromigration and self-heat limits, hot-electron rules). How these constraints are represented, how they are enforced, and what degrees of freedom (e.g., shielding, tapering, spacing, repeater insertion, driver sizing, topology design, etc.) are exploited depends on the routing tool.

We considered constraint types involving segment widths, spacings, and choice of topology. These not only are difficult to enforce within current routing approaches, but also have potentially harmful interactions with performance constraints (e.g., a watermarking constraint might require a net to be routed at minimum separation from its closest neighbors; a crosstalk constraint might dictate otherwise). We also considered various "parity" watermarking schemes based on, e.g., the orientation of the "L" for two-pin connections, the parity of the number of segments, the parity of path lengths in the routing, etc. These were dismissed as highly unnatural (e.g., pin access clearly dictates which "L" the router will choose), difficult to enforce using known routing methodologies (e.g., parity of total tree length), or vulnerable to simple tampering (e.g., tampering by compaction would ruin length-parity schemes).

Our approach bases the watermarking constraints on the (per-net) *costing* of the underlying routing resource. Specifically, for each watermark net, we impose unusual costs on "wrong-way" and/or via resources and hope that the watermark nets are provably unusual in their utilization of such resources. Many commercial routers already accept such control of the routing cost structure on a per-net basis. Our approach has the following advantages.

1) Very few constraints are needed to make a strong signature, assuming that the resource costing is reflected in the routing result for each watermark net.
2) It is compatible with many existing routing constraints, e.g., those that are based on wire width, spacing, or shielding. Potential conflicts with respect to wrong-way routing have not been an issue in our experience, particularly since we impose *upper* bounds on the use of wrong-way routing.
3) It is not easy to disturb the signature with local perturbations: the small signature size implies that many nets will need to be rerouted before the signature is likely to be unrecoverable. Furthermore, as designs are increasingly limited in terms of the interconnect resource, the routing of watermark nets is likely to be "locked in" by the routing of the remaining nonwatermark nets. Hence, destroying the watermark requires rerouting of the design.

V. Physical Design Flow With Watermarking

This section continues our detailed discussion of watermarking in physical design and proposes a concrete design flow that can be used to evaluate the strength of watermarks and their resistance to tampering. Our physical design flow uses black-box commercial tools from Cadence Design Systems: placement watermarking is built around QPlace v5.0.46 and WarpRoute v1.0.22 and routing watermarking is built around the IC Craftsman v2.1.3 router using a standard constraint type in this tool ("limit way" rule). We now give details of the experimental protocol.

A. Placement

Our experimental methodology is designed to show how easily an existing tool can be modified to offer watermarking capability. The basic comparison is shown in Fig. 2. A traditional nonwatermarked placement flow reads library and design information via library exchange format/design exchange format (LEF/DEF), executes QPlace, then executes WarpRoute to evaluate the placement quality. This is shown on the left-hand

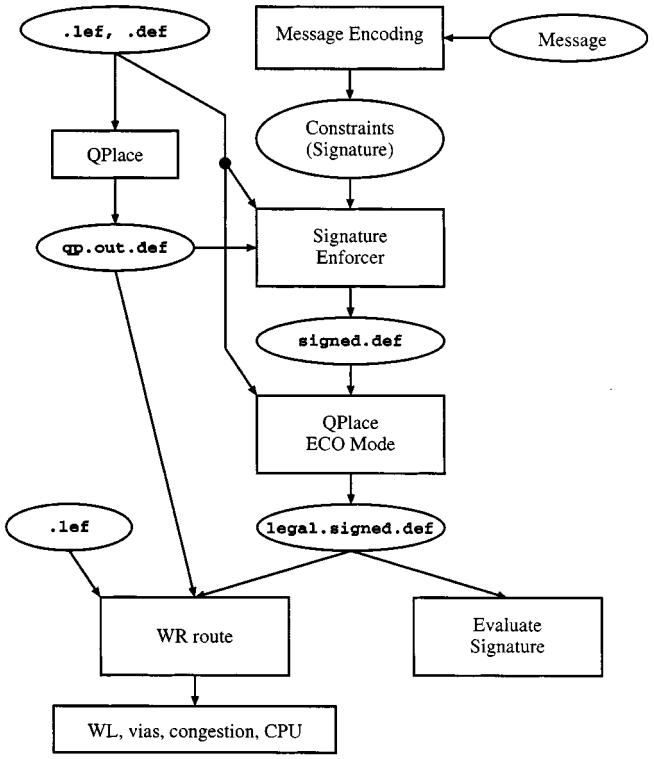


Fig. 2. Postprocessing-based watermarking protocol for standard-cell placement, using the Cadence Design Systems' QPlace v5.0.46 and WarpRoute v1.0.22 tools.

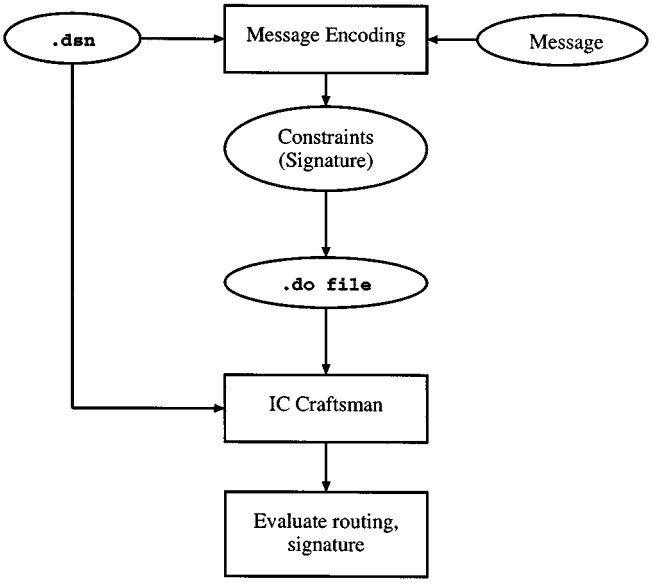


Fig. 3. Preprocessing-based watermarking protocol for (standard-cell) gridless area routing, using the Cadence Design Systems' IC Craftsman v2.1.3 tool. The tool is used in its standard context, controlled by a ".do" file using standard rules syntax.

side of Fig. 2. Our *postprocessing-based* watermarking flow, shown on the right-hand side, consists of the following steps.

1) We read the default QPlace placement result (a DEF file with location data) and the LEF file into our internal design database.
2) We ask the user for a message (e.g., "placed by QP on 10-10-97"), which we then transform into row-parity constraints for some subset of the core (nonpad, standard) cells of the design.
3) We enforce all the row-parity constraints by local changes to the placement (e.g., pair-swap operations), generating a "signed DEF" file.
4) We ensure that the resulting placement is ready for routing by rerunning in "engineering change order (ECO) mode"; this makes only minimal changes to the placement and only if necessary (typically, to avoid illegal overlaps with fixed obstacles or other cells). The output of this step is a "legal signed DEF" file.
5) We execute WarpRoute and evaluate the placement quality.

We make the following observations.

1) Our postprocessing approach is absolutely equivalent to what might be implemented in a modification of the actual commercial tool. Alternatively, our watermarking flow is trivially implemented by scripting and standard capabilities of the commercial placer (LEF/DEF manipulation, ECO placement, etc.).
2) We begin with a high-quality solution and retrospectively impose constraints. Not only is this a good approach to maintaining solution quality, but from the outside, one cannot tell whether the watermarked placement is created from scratch or by postprocessing of a nonwatermarked placement.
3) The "final list of core cells" is a well-defined concept in all existing design flows including those that invoke "in-place optimization" or "placement-based synthesis." Thus, generating a watermark based on core cell indices and row parities is also well defined.

Netlist-dependent *floorplan-independent* watermarks are also possible if a canonical row indexing is available, e.g., top-down for horizontal rows and left-right for vertical rows. One can also define the row parity constraints in terms of two equivalence classes of constrained cells. We note that the implicit assumption of unchangeable cell names can also be reasonable since any methodology allowing arbitrary renaming of cells would likely have some overhead for verification.

B. Routing

A traditional nonwatermarked routing flow using the IC Craftsman router reads library and placed design information via the .dsn file format, then executes the router under the control of a ".do file." This is shown on the left-hand side of Fig. 3. Our *preprocessing-based* watermarking flow, shown on the right-hand side of Fig. 3, consists of the following steps.

1) We identify all unique signal net names in the .dsn file.
2) We ask the user for a message (e.g., "routed by ICC on 10-10-97"), which we then transform into a list of "watermark nets" (some subset of the net names in the design).
3) We then constrain the watermark nets using IC Craftsman rules in the "do file." Specifically, our methodology applies a "limit way = 1" rule to each of the watermark nets.
4) We execute the ICC router.

C. Evaluation of Signature Strength

Each constraint involves some "random" choice, e.g., choosing a random cell or signal net. (Such choices are not actually random, but use a cryptographically strong pseudorandom number generator that is seeded with a binary signature file.) The choices may occur either with or without replacement. If there is replacement, then constraints will be independent of each other. Even if there is no replacement, the constraints may very nearly act as if they were independent, especially if the pool of constraints to choose from is large relative to the number of constraints actually chosen. As long as the constraints are either independent or nearly so, the probability P_c of a solution carrying an author's watermark purely by coincidence can be computed by a simple binomial. We use P_c to measure the strength of the 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. The probability that x or fewer out of X constraints are satisfied by coincidence is given by $P_c = \sum_{i=0}^{x} (C(X,i) \cdot (p)^{X-i} \cdot (1-p)^i)$. When constraints are not independent, the exact value of P_c may not be expressible in terms of only p and X. However, overestimating the value of p always makes P_c larger, i.e., it will weaken the estimate of the strength of our watermark and provide a useful lower bound.

1) For our placement watermarks, the signature consists of a certain subset of cells, each constrained to be in a cell row with specified-parity index. We use $p = 0.5$ as the chance that a given cell will satisfy its constraint by coincidence.
2) For our routing watermarks, the signature consists of a certain subset of signal nets, each with an "unusually low" limit on the amount of wrong-way wiring that can be used to route the net. We use the following methodology to establish a binary indicator of whether a given net has been "successfully watermarked." Given a routed design, we evaluate the total wirelength (WL_{tot}) and the wrong-way wirelength (WL_{way}) for each signal net. We then rank all nets in order of increasing value of the ratio $WL_{\text{way}}/WL_{\text{tot}}$. The watermark nets are expected to occur earlier in this ranking, while nonwatermark nets are expected to occur later in this ranking. We then establish a cutoff rank below which a watermark net is considered "successfully watermarked" and above which a watermark net is considered "not successfully watermarked." In the routing experimental results reported below, we always set the threshold rank at the 40th percentile, i.e., $p = 0.4$. (Stronger results can be obtained by more carefully choosing the value of p; this is noted below.)

D. Resistance to Tampering Attacks

Another way to evaluate the strength of a given watermark is to assess its resistance to attacks. Thus, in addition to reporting P_c values, we also report the resistance of our watermarking schemes to *tampering* attacks described below. Recall that a tampering attack attempts to remove the rightful IP owner's signature and possibly introduce the attacker's own signature into the IP. In these scenarios, the *attacker* is trying to erase the watermark by small layout perturbations.

TABLE II
NUMBER OF CELLS AND NETS IN THE SEVEN INDUSTRY TEST CASES

	Placement					Routing	
	test1	test2	test3	test4	test5	sc1	sc2
#cells	9011	12133	12857	20577	42352	1653	4250
#nets	11962	11828	10880	25634	44490	1802	1597

1) *Placement*
   a) *Assumptions:* i) The attacker has access only to an incremental ("legalizing") placement tool such as QPlace ECO mode. Recall that removing watermarking must be made as difficult as resolve the placement problem from scratch. Watermarks may not be helpful if the attacker can resolve the problem from scratch; ii) the watermarking scheme is unknown to the attacker; and iii) original design constraints are retained.
   b) *Attack:* i) Select N random pairs of cells and swap the locations of each cell pair and ii) run the legalizing placer to legalize the design (continue with routing, etc.).
2) *Routing*
   a) *Assumptions:* i) The attacker has access only to incremental (single-net) auto-routing; ii) the watermarking scheme is unknown to the attacker; and iii) original design constraints are retained.
   b) *Attack:* Select N random nets, then reroute these nets with only the original design constraints (if any).

VI. EXPERIMENTAL RESULTS

We applied our proposed physical design watermarking protocols to seven industry test cases, five in placement and two in routing. Aspects of the test cases are given in Table II. The first four placement test cases are used to assess watermark strength and the impact of our watermarking approach on various standard (postrouting) measures of placement solution quality. The last placement test case, test5, is a *timing-driven* test case that we use to confirm that the watermarking has no effect on timing quality in a timing-driven flow. The routing test case *sc2* has a relatively small number of nets relative to cells because many signals are prerouted and, hence, not included in the netlist.

A. Watermark Strength P_c

Results for the placement experiments are summarized in Table III. We report five postrouting layout quality measures for each test case. These measures are: total wirelength, total number of vias, percentage of overcongested "global routing cells" (as reported by the placer), and CPU time in (mm : ss) required by the router (all CPU times are for a 140-MHz Sun Ultra1). Together, these measures provide a fairly complete picture of the utility of each placement. In Table III, the subscript *orig* indicates the default nonwatermarked solution; the subscript $wm-x, y$ indicates a watermarked solution with x cells

TABLE III
WATERMARKING RESULTS FOR PLACEMENT

Test Case	WL	# Vias	Cong	CPU	P_c
$t1_{orig}$	6.38	86072	1.52%	11:38	
$t1_{wm-56,52}$	6.40	86595	1.52%	12:03	5.5e-12
$t1_{wm-112,96}$	6.40	86449	1.52%	12:13	2.2e-15
$t1_{wm-224,189}$	6.42	86712	1.54%	12:10	4.8e-27
$t1_{wm-448,389}$	6.44	87143	1.53%	12:08	5.7e-61
$t1_{wm-896,786}$	6.51	87716	1.52%	13:02	7.3e-127
$t1_{wm-1792,1526}$	6.62	88955	1.55%	13:25	8.3e-215
$t2_{orig}$	3.32	95601	0.86%	9:30	
$t2_{wm-56,53}$	3.33	95811	0.78%	9:20	4.1e-13
$t2_{wm-112,110}$	3.33	95978	0.80%	9:06	1.2e-30
$t2_{wm-224,208}$	3.34	95913	0.77%	9:11	4.5e-44
$t2_{wm-448,409}$	3.35	96554	0.91%	9:15	3.4e-79
$t2_{wm-896,837}$	3.38	97902	0.94%	9:28	3.2e-177
$t2_{wm-1792,1678}$	3.42	99467	1.13%	9:56	2.9e-357
$t3_{orig}$	3.13	52401	4.47%	13:58	
$t3_{wm-56,55}$	3.15	52433	4.57%	11:32	7.9e-16
$t3_{wm-112,110}$	3.14	52636	4.60%	11:45	1.2e-30
$t3_{wm-224,219}$	3.16	52529	4.65%	11:53	1.7e-58
$t3_{wm-448,443}$	3.17	53003	4.57%	11:44	2.1e-124
$t3_{wm-896,879}$	3.21	53559	5.00%	11:45	7.2e-235
$t3_{wm-1792,1740}$	3.25	54279	5.14%	11:58	3.2e-439
$t4_{orig}$	8.13	179526	0.02%	17:07	
$t4_{wm-56,50}$	8.14	179680	0.02%	17:35	5.1e-10
$t4_{wm-112,102}$	8.14	179678	0.02%	14:42	1.2e-20
$t4_{wm-224,201}$	8.16	180052	0.02%	15:45	5.7e-37
$t4_{wm-448,407}$	8.18	180590	0.02%	23:49	3.5e-77
$t4_{wm-896,797}$	8.25	182224	0.02%	19:59	1.6e-136
$t4_{wm-1792,1597}$	8.32	183783	0.02%	17:44	6.9e-274

Wirelengths are scaled to 10^6 μm.

in the signature of which y ended up being successfully watermarked.[16]

There is essentially no solution quality overhead to introducing the placement watermark. Our placement watermarking protocol also shows graceful degradation of solution quality if extremely strong signatures are required.[17]

Beyond these standard placement experiments, we have also performed an experiment with a timing-driven design flow to check the effect of our watermarking technique on timing. Table IV summarizes the results of this timing-driven experiment; we see that our watermarking technique does not have any negative effect and in fact slightly improves positive setup slack.

Results for the routing experiments are summarized in Table V. We report three postrouting layout quality measures: total wirelength (WL), total number of vias, and CPU time required by the IC Craftsman router. Since our watermarking strategy is based on limiting the length of acceptable wrong-way routing in watermarked nets, we also report total wrong-way wirelength (WW) in each solution. Finally, we report the value of P_c for each watermarked design. Increasing the signature size (i.e., the number of watermark nets constrained with the "limit way $= 1$ " rule) improves the value of P_c without significantly degrading the routing performance.

As a side note, recall from above that the value of p (a consequence of the threshold rank) may be chosen to optimize the signature strength measure P_c. Table VI shows how calculated P_c values can vary as p varies from 0.2 to 0.4. In Table VI, the second column gives the size of the signature (number of watermark nets) and each entry $x(y)$ represents the P_c value (x) and the number of unsuccessfully watermarked nets (y). We observe that fine-tuning of p (e.g., choosing $p = 0.35$) could potentially improve our results.

B. Resistance to Tampering

Tables VII and VIII present results of experiments in which we attempt to tamper with placement and routing watermarks, respectively. In each table, the second column indicates the original signature size and the third column (Init) gives the original watermarked solution quality (total WL). Subsequent columns indicate the number of cell pair-swap (net ripup and reroute) operations performed in the placement (routing) tampering, expressed as a percentage of the total number of cells (nets) in the design. We report P_c values for the 10% column to show that the watermarks remain strong even after tampering. For placement (Table VII), the solution quality degrades much faster than the signature strength, even though we restricted all random pair swaps to occur over Manhattan distances less than twice the cell row height. (ECO placement CPU times were consistent and small and we do not report them.) For routing (Table VIII), the solution quality appears relatively immune to tampering (other measures such as number of vias also remained constant). However, the CPU time required to tamper with a large number of nets approaches the cost of redoing the entire solution from scratch (at which point tampering is not needed). We conclude that our watermarking schemes are quite robust with respect to random tampering.

Finally, Fig. 4(a) shows the watermarked layout of test case *sc1* (56 watermark nets) and Fig. 4(b) shows the non-watermarked layout of the same design. We observe that it is practically impossible to notice any structural change in the watermarked solution (note that any attacker will have access only to the watermarked version of the design).

VII. CONCLUSION

Motivations and antecedents for *watermarking-based* protection of hardware and software design IP arise in reusecentric

[16]In other words, when the QPlace tool is run in ECO mode for placement legalization, some signature cells may be moved to incorrect-parity rows, which will reduce the number of cells that are successfully watermarked.

[17]We cannot assess any effects on layout area because in the modern context (three or more layers of metal with sitemap-based placement and area routing technology), the place-and-route problem is a *fixed-die* problem. In other words, the number and geometry of cell sites in cell rows are fixed before placement. In particular, all interrow spacings and track pitches are fixed. Area routers are typically used. [Routability, thus, becomes paramount: 1) congestion analysis and hot-spot removal and 2) floorplan (site map) optimization such as Cadence VSize or Avant! DSO become key parts of the place and route strategy.] The use of fixed-die approaches is driven by (hierarchical) design methodology: the presence of macros, a fixed floorplan, and fixed power and clock distribution networks together make the alternative variable-die approach less relevant. Fixed die is also driven by the process: with n-layer metal processes, blocks have high site utilization (<1% of "whitespace" is not uncommon); the use of "double-back" (shared power/ground rail) cell row architecture also fixes the row pitch. Given the fixed routing resources and block site map, any change in total wirelength (assuming the routing remains feasible) will only affect measures of congestion, not the layout area.

TABLE IV
TIMING EFFECTS OF WATERMARKING WITH 224-CELL, 448-CELL, AND 896-CELL SIGNATURE SIZES FOR THE TEST5 TIMING-DRIVEN EXAMPLE

	Original	WM – 224,217	WM – 448,403	WM – 896, 885
# Setup Violations	0	0	0	0
Min Setup Slack	1.38 ns	1.51 ns	1.51 ns	1.51 ns
Max Hold Violation	2.43 ns	2.43 ns	2.43 ns	2.43 ns
Ave Hold Violation	2.41 ns	2.41 ns	2.41 ns	2.41 ns
# Hold Violations	10	10	10	10
Routed WL (10^6 microns)	4.32	4.37	4.37	4.40
# Vias	301806	304190	304682	306104
Routing time	15:42	10:46	18:14	16:01

Watermarked solutions do not degrade the timing.

TABLE V
WATERMARKING RESULTS FOR ROUTING

Test Case	WL	WW	# Vias	CPU	P_c
$sc1_{orig}$	5.48	9.67	13440	422:27	
$sc1_{wm-20,20}$	5.46	9.56	13320	402:18	1.1e-8
$sc1_{wm-40,40}$	5.49	9.55	13426	672:19	1.2e-16
$sc1_{wm-80,78}$	5.48	9.23	13433	503:48	1.1e-28
$sc1_{wm-160,159}$	5.46	8.32	13681	641:32	5.2e-62
$sc1_{wm-320,315}$	5.47	7.51	13921	450:34	9.5e-117
$sc2_{orig}$	2.29	3.54	5590	23:52	
$sc2_{wm-20,20}$	2.29	3.60	5547	21:29	1.1e-8
$sc2_{wm-40,40}$	2.30	3.77	5716	18:58	1.2e-16
$sc2_{wm-80,79}$	2.30	3.47	5713	16:50	1.8e-30
$sc2_{wm-160,151}$	2.29	3.23	5650	20:43	1.3e-48
$sc2_{wm-320,294}$	2.30	2.93	5706	20:43	2.2e-85

Total wirelengths (WL) are scaled to 10^7 user database units and wrong-way wirelengths (WW) are scaled to 10^6 units.

TABLE VI
P_c VALUES CORRESPONDING TO DIFFERENT VALUES OF p

test case	# of nets	p values 0.2	0.25	0.3	0.35	0.4
sc1	20	1.1e-14(0)	9.1e-13(0)	3.5e-11(0)	7.6e-10(0)	1.1e-8(0)
	40	1.4e-24(2)	1.0e-22(1)	1.2e-21(0)	5.8e-19(0)	1.2e-16(0)
	80	3.0e-46(5)	8.9e-41(4)	2.6e-38(2)	3.7e-33(2)	1.1e-28(2)
	160	3.5e-91(10)	4.9e-82(7)	1.7e-75(4)	3.4e-71(1)	5.2e-62(1)
	320	6.2e-91(88)	2.2e-115(46)	3.2e-129(20)	8.9e-124(11)	9.5e-117(5)
sc2	20	1.5e-5(7)	1.6e-9(2)	3.8e-8(2)	7.6e-10(0)	1.1e-8(0)
	40	7.9e-10(14)	2.5e-15(6)	5.3e-18(2)	4.4e-17(1)	1.2e-16(0)
	80	1.9e-13(34)	7.0e-32(10)	2.5e-33(5)	3.7e-33(2)	1.8e-30(1)
	160	2.6e-17(80)	5.8e-32(49)	3.3e-47(24)	2.3e-45(18)	1.3e-48(9)
	320	2.5e-8(214)	2.4e-34(137)	7.7e-61(81)	1.5e-83(41)	2.2e-85(26)

Each entry $x(y)$ represents the P_c value (x) and the number of unsuccessfully watermarked nets (y).

TABLE VII
RESULT FROM TAMPERING WITH THE PLACEMENT WATERMARK

Test	# cells	Init	1 %	2 %	5 %	10 %	P_c
t3	56	3.15	3.24	3.34	3.61	4.00	4.1e-13
	112	3.14	3.25	3.33	3.60	4.03	4.9e-25
	224	3.16	3.26	3.34	3.61	4.02	4.5e-44
	448	3.17	3.27	3.36	3.62	4.04	9.1e-95
	896	3.21	3.30	3.39	3.65	4.08	1.1e-180
	1792	3.25	3.35	3.44	3.69	4.12	1.1e-334
t4	56	8.14	8.28	8.68	9.83	11.57	1.3e-7
	112	8.14	8.31	8.75	9.80	11.52	1.3e-14
	224	8.16	8.34	8.71	9.86	11.64	2.7e-29
	448	8.18	8.36	8.76	9.80	11.62	3.4e-48
	896	8.25	8.42	8.88	9.96	11.63	6.2e-94
	1792	8.32	8.48	8.88	9.94	11.64	6.1e-196

Solution quality degrades much faster than signature strength. Hence, tampering does not appear to be a viable form of attack.

TABLE VIII
RESULT FROM TAMPERING WITH THE ROUTING WATERMARK

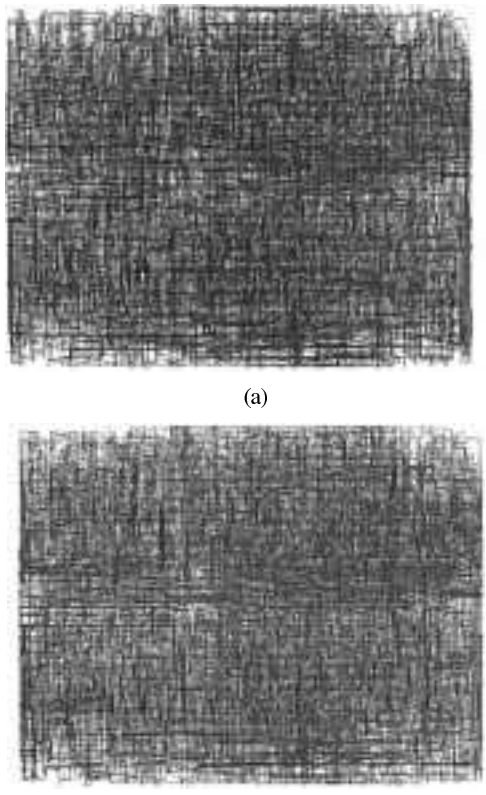
Test	# nets	Init	1%	2%	5%	10%	P_c
sc1	20	5.46	5.47	5.47	5.49	5.53	1.1e-5
	40	5.49	5.50	5.50	5.51	5.55	1.3e-11
	80	5.48	5.48	5.48	5.49	5.54	4.2e-21
	160	5.46	5.46	5.47	5.48	5.54	3.1e-38
	320	5.47	5.48	5.48	5.49	5.54	1.4e-61
	CPU	534:06	28:30	51:58	118:26	127:25	
sc2	20	2.29	2.29	2.29	2.29	2.30	7.7e-8
	40	2.30	2.30	2.30	2.31	2.31	8.6e-13
	80	2.30	2.29	2.30	2.30	2.30	1.1e-23
	160	2.29	2.29	2.29	2.30	2.30	2.3e-36
	320	2.30	2.30	2.30	2.31	2.31	1.7e-67
	CPU	19:45	1:24	2:18	3:57	8:39	

CPU times for ripup and reroute approach the time required to resolve the problem from scratch. Hence, tampering does not appear to be a viable form of attack.

system design, artifact watermarking, and cryptography. In this paper, we have described fundamental precepts, a canonical technique, and example applications for watermarking-based IPP. Several key ideas are as follows.

1) Stages of the (hardware, software) design process can typically be viewed as (difficult) *optimization instances* whose solutions constitute IP to be protected.
2) IP watermarking can typically be achieved by adding *constraints* (e.g., interpreted from a cryptographically secure encoding of the IP owner's signature) to any given design optimization instance.
3) The addition of constraints can typically be achieved using *pre-* or *postprocessing* of the inputs and outputs, respectively, for a given design optimization. In this way, the watermarking is often transparent to existing algorithms and tools, i.e., it is *nonintrusive*.

We have also noted other aspects of the watermarking context, e.g., protection requirements against typical forms of attack and cryptography background (one-way functions, cipher streams, and digital signatures). Problem formulations from several domains (high-level design, FPGA design, physical design, as well as SAT) illustrate the general applicability of our techniques and suggest that nonintrusive IP watermarking with constraints can typically be implemented with no significant added complexity or loss of solution quality. Thus, constraint-based



(a) (b)

Fig. 4. Routing solution. (a) Watermarked and (b) nonwatermarked for the sc1 test case.

watermarking appears to have significant potential to protect IP and support design reuse.

In addition, we have developed the first IPP protocols for embedding design watermarks at the physical-design level. We have implemented these protocols transparently to existing design flows, using leading industrial tools. On real designs, we show strong proofs of authorship with very acceptable cost overhead for the watermarking and no impact on layout area (given the fixed-die context) or timing. We also show the robustness of our watermarking scheme with respect to random tampering attacks.

Our ongoing work develops watermarking-based IPP techniques for many other domains with particular attention to robustness under various attacks. We also address a number of variant requirements, including fingerprinting, copy detection, and proportionate watermarking (e.g., of hierarchical designs).

REFERENCES

[1] C. J. Alpert, T. Chan, D. J. Huang, A. B. Kahng, I. Markov, P. Mulet, and K. Yan, "Faster minimization of linear wirelength for global placement," in *Proc. ACM/IEEE Int. Symp. Physical Design*, Apr. 1997, pp. 4–11.

[2] C. J. Alpert and A. B. Kahng, "Recent directions in netlist partitioning: A survey," *Integr. VLSI J.*, vol. 19, no. 1-2, pp. 1–81, Aug. 1995.

[3] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for data hiding," *IBM Syst. J.*, vol. 35, no. 3–4, pp. 313–336, 1996.

[4] H. Berghel and L. O'Gorman, "Protecting ownership rights through digital watermarking," *IEEE Comput.*, vol. 29, pp. 101–103, July 1996.

[5] K. D. Boese, A. B. Kahng, and S. Muddu, "New adaptive multistart techniques for combinatorial global optimizations," *Oper. Res. Lett.*, vol. 16, no. 2, pp. 101–113, 1994.

[6] L. Boney, A. H. Tewfik, and K. N. Hamdy, "Digital watermarks for audio signals," in *Proc. 3rd IEEE Int. Conf. Multimedia Computing and Systems*, June 1996, pp. 473–480.

[7] A. G. Bors and I. Pitas, "Image watermarking using DCT domain constraints," in *Proc. Int. Conf. Image Processing*, vol. 3, Sept. 1996, pp. 231–344.

[8] J. T. Brassil, S. Low, N. F. Maxemchuk, and L. O'Gorman, "Electronic marking and identification techniques to discourage document copying," *IEEE J. Select. Areas Commun.*, vol. 13, pp. 1495–1504, Apr. 1995.

[9] J. Brassil and L. O'Gorman, "Watermarking document images with bounding box expansion," in *Proc. First Int. Workshop Information Hiding*, May 1996, pp. 227–235.

[10] G. W. Braudaway, K. A. Magerlein, and F. Mintzer, "Protecting publicly available images with a visible image watermark," *Proc. SPIE-Int. Soc. Opt. Eng.*, vol. 2659, pp. 126–133, 1996.

[11] R. E. Bryant, "Binary decision diagrams and beyond: Enabling technologies for formal verification," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1995, pp. 236–243.

[12] S. T. Chakradhar, V. D. Agrawal, and S. G. Rothweiler, "A transitive closure algorithm for test generation," *IEEE Trans. Computer-Aided Design*, vol. 12, pp. 1015–28, July 1993.

[13] E. Charbon, "Hierarchical watermarking in IC design," in *Proc. IEEE Custom Integrated Circuits Conf.*, May 1998, pp. 295–298.

[14] E. Charbon and I. Torunoglu, "Watermarking layout topologies," in *Proc. Asia South Pacific Design Automation Conf.*, Jan. 1999, pp. 213–216.

[15] D. C.-M. Chi, "Improving upon local search heuristics for VLSI standard cell placement," M.S. thesis, Comput. Sci. Dept., Univ. California, Los Angeles, CA, 1995.

[16] I. J. Cox, J. Kilian, T. Leighton, and T. Shamoon, "Secure spread spectrum watermarking for images, audio and video," in *Proc. Int. Conf. Image Processing*, vol. 3, Sept. 1996, pp. 243–246.

[17] I. J. Cox and M. L. Miller, "A review of watermarking and the importance of perceptual modeling," in *Proc. SPIE Conf. Human Vision and Electronic Imaging II*, vol. 3016, 1997, pp. 92–99.

[18] S. Craver, N. Memon, B.-L. Yeo, and M. Yeung, "Can invisible watermarks resolve rightful ownerships?," in *Proc. SPIE Storage and Retrieval for Images and Video Databases V*, vol. 3022, Feb. 1997, pp. 310–321.

[19] M. Davis and H. Putnam, "A computing procedure for quantification theory," *J. ACM*, vol. 7, no. 3, pp. 201–215, 1960.

[20] J.-F. Delaigle, C. De Vleeschouwer, and B. Macq, "Digital watermarking," in *Proc. SPIE Optical Security and Counterfeit Deterrence Techniques*, vol. 2659, Feb. 1996, pp. 99–110.

[21] S. Devadas, "Optimal layout via Boolean satisfiability," in *IEEE Int. Conf. Computer-Aided Design*, Nov. 1989, pp. 294–297.

[22] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 644–654, Nov. 1976.

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

[24] J. Giraldi and M. L. Bushnell, "Search state equivalence for redundancy identification and test generation," in *Proc. Int. Test Conf.*, Oct. 1991, pp. 184–193.

[25] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *J. ACM*, vol. 42, no. 6, pp. 1115–45, 1995.

[26] F. Goffin *et al.*, "A low cost perceptive digital picture watermarking method," in *Proc. SPIE Storage and Retrieval for Image and Video Databases V*, vol. 3022, Feb. 1997, pp. 264–277.

[27] F. Hartung and B. Girod, "Copyright protection in video delivery networks by watermarking of precompressed video," in *Proceedings of the Second European Conference on Multimedia Applications, Services, and Techniques*, S. Fdida and M. Morganti, Eds. Berlin, Germany: Springer-Verlag, 1997, vol. 1242, Lecture Notes in Computer Science, pp. 423–436.

[28] ——, "Watermarking of MPEG-2 encoded video without decoding and re-encoding," in *Proc. Multimedia Computing and Networking*, Feb. 1997, pp. 264–274.

[29] A. H. Hashemi, D. R. Kaeli, and B. Calder, "Efficient procedure mapping using cache line coloring," *ACM SIGPLAN Notices*, vol. 32, no. 5, pp. 171–182, 1997.

[30] I. Hong and M. Potkonjak, "Techniques for intellectual property protection of DSP designs," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, vol. 5, May 1998, pp. 3133–3136.
[31] L. Honey, A. H. Tewfik, and K. N. Hamdy, "Digital watermarks for audio signals," in *Proc. 3rd IEEE Int. Conf. Multimedia Computing and Systems*, June 1996, pp. 473–480.
[32] H. T. Jankowitz, "Detecting plagiarism in student PASCAL programs," *Comput. J.*, vol. 31, no. 1, pp. 1–8, 1988.
[33] S. A. Kauffman, *At Home in the Universe: The Search for Laws of Self-Organization and Complexity*. New York: Oxford Univ. Press, 1995.
[34] D. Kirovski and M. Potkonjak, "System-level synthesis of low-power hard real-time systems," in *Proc. ACM/IEEE Design Automation Conf.*, June 1997, pp. 697–702.
[35] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "Fingerprinting digital circuits on programmable hardware," in *Proc. Workshop Information Hiding*, Apr. 1998, pp. 16–31.
[36] ——, "FPGA fingerprinting techniques for protecting intellectual property," in *Proc. IEEE Custom Integrated Circuits Conf.*, May 1998, pp. 299–302.
[37] T. Larrabee, "Test pattern generation using boolean satisfiability," in *Proc. Int. Test Conf.*, Aug. 1992, pp. 4–15.
[38] S. H. Low, N. F. Maxemchuk, J. T. Brassil, and L. O'Gorman, "Document marking and identification using both line and word shifting," in *Proc. IEEE Infocom*, Apr. 1995, pp. 853–60.
[39] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, S. A. Vanstone, Ed. Boca Raton, FL: CRC, 1997.
[40] A. Parker and J. O. Hamblen, "Computer algorithms for plagiarism detection," *IEEE Trans. Educ.*, vol. 32, pp. 94–99, May 1989.
[41] *Pretty Good Privacy (PGP) User's Guide*, vol. 1, 2, Phil's Pretty Good Software (Pretty Good Privacy, Inc.), San Mateo, CA.
[42] G. Polya, R. E. Tarjan, and D. R. Woods, *Notes on Introductory Combinatorics*. Cambridge, MA: Birkhaüser, 1983.
[43] J. Riordan, *An Introduction to Combinatorial Analysis*. Princeton, N.J.: Princeton Univ. Press, 1980.
[44] R. L. Rivest, "RFC 1321: The MD5 Message-Digest Algorithm," Internet Activities Board, 1992.
[45] B. Schneier, *Applied Cryptography : Protocols, Algorithms and Source Code in C*, 2nd ed. New York: Wiley, 1996.
[46] B. Selman, "Stochastic search and phase transitions: AI meets physics," *Proc. Int. Joint Conf. Artificial Intelligence*, vol. 1, pp. 998–1002, Aug. 1995.
[47] *National Technology Roadmap for Semiconductors*, Semiconductor Industry Association, San Jose, CA, 1997.
[48] G. A. Spanos and T. B. Maples, "Performance study of a selective encryption scheme for the security of networked, real-time video," in *Proc. Int. Conf. Computer Communications and Networks*, Sept. 1995, pp. 2–10.
[49] D. R. Stinson, *Cryptography: Theory and Practice*. Boca Raton, FL: CRC, 1995.
[50] M. D. Swanson, B. Zhu, and A. H. Tewfik, "Transparent robust image watermarking," in *Proc. Int. Conf. Image Processing*, vol. 3, Sept. 1996, pp. 211–214.
[51] W. Swartz and C. Sechen, "Timing driven placement for large standard cell circuits," in *Proc. ACM/IEEE Design Automation Conf*, June 1995, pp. 211–215.
[52] G. E. Tellez, D. A. Knol, and M. Sarrafzadeh, "A performance-driven placement technique based on a new budgeting criterion," in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. 4, May 1996, pp. 504–507.
[53] A. H. Tewfik and M. Swanson, "Data hiding for multimedia personalization, interaction and protection," *IEEE Signal Processing Mag.*, vol. 14, pp. 41–44, July 1997.
[54] A. Z. Tirkel, C. F. Osborne, and R. G. Van Schyndel, "Image watermarking—a spread spectrum application," in *Proc. Int. Symp. Spread Spectrum Techniques and Applications*, vol. 2, Sept. 1996, pp. 785–789.
[55] S. Trimberger, private communication, 1997.
[56] "Fall Worldwide Member Meeting: A Year of Achievement (Guidelines Proposed by VSIA Development Working Group on Intellectual Property Protection)," VSI Alliance, Santa Clara, CA, 1997.
[57] H. Wang, T. Sun, and Q. Yang, "Minimizing area cost of on-chip cache memories by caching address tags," *IEEE Trans. Comput.*, vol. 46, pp. 1187–1201, Nov. 1997.
[58] *The Programmable Logic Data Book*, Xilinx Corp., San Jose, CA, 1996.

Andrew B. Kahng was born in San Diego, CA, in October 1963. He received the A.B. degree in applied mathematics/physics from Harvard College, Cambridge, MA, and the M.S. and Ph.D. degrees in computer science from the University of California at San Diego, La Jolla.

He was with the Computer Science Department, University of California, Los Angeles, from 1989 to 2000, most recently as Professor and Vice-Chair. He is currently a Professor of Computer Science and Engineering and Electrical and Computer Engineering, University of California at San Diego. He has authored or coauthored over 160 papers in the VLSI computer-aided design literature, centering on physical layout and performance analysis. His current research interests include VLSI physical layout design and performance analysis, combinatorial and graph algorithms, and stochastic global optimization.

Prof. Kahng received the National Science Foundation Young Investigator Award and a Design Automation Conference Best Paper Award. He was the founding General Chair of the ACM/IEEE International Symposium on Physical Design, cofounder of the ACM Symposium on System-Level Interconnect Prediction, and has defined the physical design roadmap for the Semiconductor Industry Association International Technology Roadmap for Semiconductors (ITRS) since 1997. He is currently Chair of the U.S. Design Technical Working Group for the 2001 and the Technical Program Chair of the 2001 Electronic Design Processes Symposium of the IEEE Design Automation and Technical Committee. He is also on the steering committees of the 2001 International Symposium on Physical Design and the 2001 International Workshop on System-Level Interconnect Prediction.

John Lach (S'99–M'00) received the B.S. degree in science, technology, and society from Stanford University, Stanford, CA, in 1996 and the M.S. and Ph.D. degrees in electrical engineering from the University of California, Los Angles (UCLA), in 1998 and 2000, respectively.

Since 2000, he has been an Assistant Professor in the Electrical and Computer Engineering Department, University of Virginia, Charlottesville. He is the Principal Investigator on the dynaptable computing project funded by the National Science Foundation. His current research interests include real-time embedded systems, computer-aided design techniques for very large scale integration, general-purpose and application-specific processor design, intellectual property protection, and field programmable gate arrays.

Dr. Lach received the 2001-2002 University of Virginia Teaching Fellowship and two School of Engineering and Applied Sciences Dean's Awards from UCLA.

William H. Mangione-Smith (M'95) received the B.S.E. degree in electrical engineering and the M.S.E. and Ph.D. degrees in computer science and engineering from the University of Michigan, Ann Arbor, in 1987, 1992, and 1992, respectively.

From 1991 to 1995, he was with Motorola, Inc., where he participated in the design of the Envoy Personal Digital Assistant. In 1995, he joined the Electrical Engineering Department at the University of California, Los Angeles, where he is currently an Associate Professor. He is a Co-Principal Investigator on the Mojave configurable computing program and served as the Program Chair for MICRO 26. His current research interests include using dynamic circuits to implement configurable computing systems, low-power processor and system design, multimedia and communications processing, and all techniques for leveraging instruction-level parallelism.

Dr. Mangione-Smith is a member of the ACM. He received the National Science Foundation CAREER Award in 1998.

Stefanus Mantik (S'01) was born in Jakarta, Indonesia, in 1974. He received the B.S. degree in information and computer science from the University of California, Irvine, in 1996 and the M.S. degree in computer science from the University of California, Los Angeles, in 1998. He is currently working toward the Ph.D. degree in computer science at the same university.

He is the Publicity Chair of the 2001 Electronic Design Processes Symposium of the IEEE Design Automation and Technical Committee. His current research interests include VLSI design process analysis and optimization, design frameworks, incremental algorithms, and distributed design environments.

Mr. Mantik is a student member of the ACM.

Igor L. Markov received the undergraduate degree in mathematics from Kiev University, Kiev, Ukraine, in 1993 and the M.A. degree in pure mathematics and the Ph.D. degree in computer science from the University of California, Los Angeles, in 1994 and 2000, respectively.

He is currently an Assistant Professor of Computer Science and Electrical Engineering, University of Michigan, Ann Arbor. In 1995, he was with the Parametric Technology Corporation, Waltham, MA, working on solid modeling computer-aided design software.

Dr. Markov received the best Ph.D. student award from the University of California, Los Angeles, in 2000.

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

In 1991, he joined the Computer and Communicatons Research Laboratories, NEC USA, Princeton, NJ. He joined the University of California, Los Angeles (UCLA), in 1995, where he is currently a Professor with the Computer Science Department. He has served on a number of program committees, including the International Conference on Image Processing, the Design Automation Conference, and the International Conference on Computer-Aided Design. He has authored or coauthored approximately 200 papers in journals and conferences and holds five patents. His recent watermarking-based intellectual property protection (IPP) research formed a basis for the Virtual Socket Initiative Alliance (IPP) standard. His current research interests include embedded systems, sensor networks, computational security, and intellectual property protection.

Dr. Potkonjak received the National Science Foundation CAREER Award, the Okawa Foundation Award, UCLA TRW SEAS Excellence in Teaching Award, and a number of best paper awards.

Paul Tucker received the Ph.D. degree in computer science from the University of California at San Diego, La Jolla, in 1999.

He is currently with Digital Integrity, Inc., San Mateo, CA.

Huijuan Wang received the B.S. degree in chemistry from the University of Science and Technology of China, Hefei, in 1995 and the M.S. degree in computer science from the University of California, Los Angeles, in 1998

She is currently a Member of Technical Staff at Sun Microsystems, Inc., Palo Alto, CA. Her current research interests include interconnect parasitic extraction and signal electromigration analysis methodology.

Gregory Wolfe, photograph and biography not available at the time of publication.