Watermarking Techniques for Intellectual Property Protection*

A. B. Kahng, J. Lach[†], W. H. Mangione-Smith[†], S. Mantik, I. L. Markov,
M. Potkonjak, P. Tucker[‡], H. Wang and G. Wolfe

UCLA Computer Science Dept., Los Angeles, CA 90095-1596
(†) UCLA Electrical Engineering Dept., Los Angeles, CA 90095-1594
(‡) UCSD Computer Science & Engineering Dept., La Jolla, CA 92093-0114

Abstract

Digital system designs are the product of valuable effort and know-how. Their embodiments, from software and HDL 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 (i) nearly invisible to human and machine inspection, (ii) difficult to remove, and (iii) permanently embedded as an integral part of the design. We survey related work in cryptography and design methodology, then develop desiderata, metrics and example approaches – centering on *constraint-based* techniques – for watermarking at various stages of the VLSI design process.

1 Introduction

The advance of processing technology has led to a rapid increase in 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 multi-billion 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" [26], 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 IP companies, third-party ASIC libraries, tools for IP integration, and industry organizations such as the VSI Alliance [33] 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.[2]

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 work, we establish principles for development of new *watermarking-based* IPP procedures. These principles, which are centered around the use of *constraints* to "sign" the output of a given design synthesis or optimization, are compatible with current IP development tools infrastructure. Furthermore, our principles apply to the protection of both hardware and software (e.g., Verilog or C++ code) IP.

A Motivating Example: 3SAT

We illustrate key ideas behind watermarking-based IPP using the *satisfiability* (SAT) problem [10]:

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 [10]. Problems from many application domains have been modeled as SAT instances. In VLSI CAD, SAT formulations have been used in testing [4, 8, 11, 20], logic synthesis and physical design [8].

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:

$$U = \{u_1, u_2, \ldots, u_{14}\}$$

$$\begin{aligned} C = \{ & \{\bar{u}_1\bar{u}_2u_9\}, \{\bar{u}_1\bar{u}_3\bar{u}_4\}, \{\bar{u}_1u_2\bar{u}_5\}, \{u_1\bar{u}_2u_{10}\}, \{\bar{u}_1\bar{u}_3u_8\}, \{\bar{u}_1\bar{u}_3u_7\}, \\ & \{u_1\bar{u}_5u_7\}, \{\bar{u}_1\bar{u}_6\bar{u}_{12}\}, \{\bar{u}_1u_{10}u_{12}\}, \{\bar{u}_1u_6u_9\}, \{\bar{u}_2\bar{u}_3\bar{u}_{10}\}, \{u_2\bar{u}_5\bar{u}_{14}\}, \\ & \{\bar{u}_2u_7u_8\}, \{u_2\bar{u}_8u_9\}, \{u_3u_4u_8\}, \{u_3u_5\bar{u}_7\}, \{\bar{u}_3u_8u_{13}\}, \{u_3\bar{u}_9\bar{u}_{11}\}, \\ & \{u_3u_{10}\bar{u}_{12}\}, \{\bar{u}_4\bar{u}_7\bar{u}_8\}, \{\bar{u}_5\bar{u}_8\bar{u}_{12}\}, \{u_4\bar{u}_7u_{13}\}, \{\bar{u}_5\bar{u}_9\bar{u}_{11}\}, \{\bar{u}_5u_7u_9\}, \\ & \{u_6u_{10}u_{11}\}, \{u_6\bar{u}_8\bar{u}_{12}\}, \{u_7u_9\bar{u}_{12}\}, \{u_7u_9\bar{u}_{13}\}, \{u_9u_{11}\bar{u}_{14}\}, \{u_{10}u_{11}\bar{u}_{12}\}\} \end{aligned}$$

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

* Work by M. Potkonjak and G. Wolfe supported in part by DARPA under grant N66001-97-2-8901. Work by W. H. Mangione-Smith and J. Lach supported by the Defense Advanced Research Projects Agency of the United States of America, under contract DAB763-95-C-0102 and subcontract QS5200 from Sanders, a Lockheed Martin company. Work by A. B. Kahng, S. Mantik, I. L. Markov, P. Tucker and H. Wang supported by a grant from Cadence Design Systems, Inc.

[1] According to [26], 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] For example, the Virtual Socket Interface (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, intellectual property protection is considered to be a crucial enabling technology [29].

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 3-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.[3] We choose the signature "Watermarking Techniques for Intellectual Property Protection University of California at Los Angeles VLSI CAD LAB", which adds 38 new clauses to the instance. After adding these constraints to the initial instance, the number of satisfying assignments decreases to 2. 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 2 in 556, or 0.00496. In this example, the addition of a watermark incurs no overhead; it simply guides which solution is selected. The watermarking strategy is also *non-intrusive* (more specifically, it is based on *pre-processing* of the input instance), in that any existing solution strategy remains applicable to the augmented (watermarked) 3SAT instance.[4] In our experience, many commonly encountered NP-complete formulations can also be watermarked using similar constraints.

1.1 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 2. Principles and desiderata (e.g., protection requirements) of non-intrusive, constraint-based IP watermarking are discussed in Section 3. A review of cryptography background and supporting tools (e.g., one-way functions, pseudo-random encrypted streams, digital signatures, and strength analyses) is given in Section 4. Section 5, as well as the above discussion in Section 1, suggests that non-instrusive (pre- or post-processing based) IP watermarking with constraints is often easy to implement with no significant added complexity or loss of solution quality. While an analysis of typical attacks in Section 4 shows that not all possible protection levels can be achieved with known algorithms, we nonetheless conclude in Section 6 that constraint-based watermarking has significant potential to protect IP and support design reuse.

2 Related Work

We now survey related work in watermarking-based IPP, cryptography, and IP-based synthesis.

2.1 Artifact vs. IP Protection Watermarking

A *watermark* is a mark that is (i) embedded into an artifact (text, image, video, audio) or piece of intellectual property (hardware, software, algorithm, data organization), (ii) designed to identify the author, the source, the used tools and techniques and/or recipient of the artifact or the intellectual property, and (iii) difficult to detect and remove. 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.[5]

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 [15], behavior [14], logic synthesis and physical design [16], as well as in FPGA designs [18, 19].

2.2 Cryptography

Modern age cryptography grew from the seminal work of Diffie and Hellman [9], who introduced public-key cryptography based on the computational intractability of certain mathematical tasks. 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. [27, 22]. The link between cryptography and watermarking is fundamental: 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.

2.3 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.[6]

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) [32], SI^2 [34] (ASIC vendors), and VSIA [33] (a large organization of EDA vendors, ASIC vendors, system houses and IP providers) – are actively building the industry infrastructure for IP-based design.[7] Several missing infrastructure pieces are technical, with deep implications for the associated EDA technology and design methodologies.[8] 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.

3 Precepts and an Approach for 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 non-intrusive watermarking (i.e.,

[3]For example, the signature "cat dog fox" would be encoded using the extra clauses $\{\{u_2, u_1, \bar{u}_{10}\}, \{u_{14}, \bar{u}_2, u_8\}, \{u_4, u_{14}, \bar{u}_3\}, \{u_8, \bar{u}_{12}, u_{14}\}\}$ (we pad the end of the message with an extra space to maintain three literals per clause).

[4]This observation holds for the three major classes of SAT heuristics: (i) random search [25, 7], (ii) nonlinear programming relaxation and rounding [12], and (iii) a variety of BDD-based techniques [3].

[5]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 [1, 15], text [21, 2], image [5], and video.

[6]Interestingly, the vision of pervasive IP reuse can be viewed as simply a refinement of earlier visions which saw the inability of the structured-custom methodology to scale with design complexity as a main driver for "methodology convergence".

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

[8]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.

methods that can be transparently integrated within existing design flows via pre- or post-processing).

3.1 Context for Watermarking

The following ingredients form the *context* for a non-intrusive watermarking procedure:

- 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.
- A well-defined *interpretation* of the solutions of the optimization problem as intellectual property.
- 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 pre- and post-processing stages.[9]
- *Protection requirements* that are largely similar to well-understood protection requirements for currency watermarking. Examples of such requirements, which are discussed in Section 4 below, include: (i) removing or forging a watermark must be as hard as re-creating the design; and (ii) tampering with a watermark must be provable in court.

A non-intrusive 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:

- 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.
- Algorithmic descriptions of the *pre-* and *post-processing* steps of the watermarking procedure.
- *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.
- *General robustness analyses*, including discussion of susceptibility to typical attacks, discussion of possible new 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: (i) having multiple acceptable solutions (we typically accept suboptimal solutions for NP-hard problems); (ii) solved by optimization heuristics; and (iii) discrete in nature.[10]

[9] Watermarking the results of non-deterministic and/or unknown algorithms – or even "hand-made" 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.

3.2 General Strategy for Constraint-Based IPP

Our general strategy is to map an author's signature into a set of constraints ("desired relations") which 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., pre- or post-processing) 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).

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 using 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.

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.

4 Analysis of Constraint-Based Watermarking

We now analyze the constraint-based watermarking strategy and present the cryptographic background and support tools that are necessary for the analysis. We first describe how to map a signature into a set of constraints and the method by which we determine the strength of the watermark in a watermarked solution. We then discuss typical forms of attack on our scheme and the obstacles that prevent these attacks from succeeding.

4.1 Selection of Constraints

Given a pseudorandom number generator and a particular type of constraint it is a simple matter to select a set of X constraints where each one is determined independently. It is only slightly more work to select a set of X constraints with no constraint repeated. 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. This is also easy. 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 (using a one-way hash function such as MD5 [24]) and using the hash as a seed for a stream cipher such as RC4 [35].

[10] We believe "continuous watermarking" is possible as well, e.g., by mapping into "discrete watermarking" by Fourier transform. However, watermarking has traditionally been of more relevance to discrete problems.

4.2 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 non-watermarked 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 $P_c \equiv P(X \leq b) = \sum_{i=0}^{b}(\frac{C!}{(C-i)!\cdot i!} \cdot (p)^{C-i} \cdot (1-p)^i)$. 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.

4.3 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.

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 *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. 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., $P_c \leq 2^{-56}$).

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 re-solving the problem from scratch with his own watermark encoded, then continuing 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 re-solve 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 (i) local perturbations of the solution to the watermarked phase, or to (ii) global-scale transformations such as those which 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.[11]

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 [23]. 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 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.

5 IP Watermarking Synthesis Examples

In this section, we sketch three examples of IP watermarking approaches, in three very distinct domains. Our intent is to illustrate the wide-ranging applicability of the principles developed above.

5.1 Preprocessing in System-Level Design

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 [17]. Much effort has been devoted to allocating minimal cache structures and optimizing code for effective cache utilization [30]. A particularly successful technique is *cache line coloring* [13].

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).[12]

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 [14], 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.

[11] 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 (at which point the tampered solution ceases to be useful); see [16] for experience in the physical design realm. However, it seems possible for an attacker to use tampering methods to remove a signature that is known to him, or to add an entirely new signature.

[12] Kirovski et al. [17] have experimentally shown that this optimization results in significant performance increase. In general, such optimizations can play an important role in the design of modern multimedia, communications, or low-power systems-on-silicon.

5.2 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 pre-processing, iterative, or post- processing techniques. The advantage of post-processing is that it does not impact other CAD design tools, and has zero impact on design peformance, 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. [18, 19].

An example of a purely post-processing 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 [31]. 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 2-to-1 multiplexer, and the two sequential outputs are each controlled by three 2-to-1 multiplexers and one 4-to-1 multiplexer. Figure 1 shows the control layout of the 4000 family's CLB outputs.

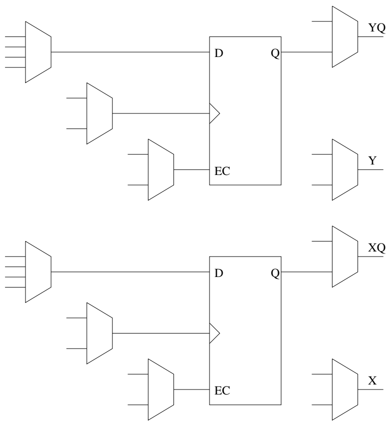

Figure 1: Control directly attributed to CLB outputs.

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 2-to-1 and 4-to-1 multiplexer respectively. Thus, each unused combinatorial output can store one watermark bit and each unused sequential output can store five watermark bits. The total number of watermark bits that can be inserted in an entirely unused CLB is twelve. Table 1 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 post-processing 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.

% 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

Table 1: Number of bits available for watermarking.

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 [28].

5.3 Preprocessing in Physical Design

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.

- physical floorplan, library of physical cell masters, and cell-level netlist
- cell-level performance macromodels for each cell master (e.g., non-linear table models (Synopsys .lib, Cadence .ctlf, OVI ALF, etc.)) for timing and power dissipation analysis
- technology file (models of interconnect RC characteristics, design rules, etc.)
- constraints, which are chiefly (i) "direct" placement and routing constraints (e.g., region-based location constraints arising from the floorplanner, and transmitted in PDEF format), and (ii) 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 $t(C_1 - C_2 - C_3 - \ldots - C_{10}) \leq 50ns$ ($C_i \equiv$ cells). We can allocate the timing bound between two subpaths and replace this constraint by two constraints $t(C_1 - \ldots - C_5) \leq 20ns$ and $t(C_5 - \ldots - C_{10}) \leq 30ns$. All else being equal, the chance that satisfying the original constraint happens to satisfy both of these subpath constraints is at most one-half.[13] Constraining on

[13] 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.

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.[14]

6 Conclusions

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

- Stages of the (hardware, software) design process can typically be viewed as (difficult) *optimization instances* whose solutions constitute intellectual property to be protected.
- 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.
- The addition of constraints can typically be achieved using *pre-* or *post-processing* 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 *non-intrusive*.

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 non-instrusive IP watermarking with constraints can typically be implemented with no significant added complexity or loss of solution quality. Thus, constraint-based watermarking appears to have significant potential to protect IP and support design reuse.

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] W. Bender, D Gruhl, N. Morimoto and A. Lu, "Techniques for Data Hiding", *IBM Systems Journal*, 35(3-4), 1996, pp. 313-336.

[2] J.T. Brassil, S. Low, N.F. Maxemchuk and L. O'Gorman, "Electronic Marking and Identification Techniques to Discourage Document Copying", *IEEE Journal on Selected Areas in Communications*, 13(4) (1995), pp. 1495-1504.

[3] R.E. Bryant, "Binary Decision Diagrams and Beyond: Enabling Technologies for Formal Verification", *Proc. of the International Conference on Computer-Aided Design*, 1995, pp. 236-243.

[4] S.T. Chakradhar, V.D. Agrawal and S.G. Rothweiler, "A Transitive Closure Algorithm for Test Generation", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12(7) (1993), pp.1015-28.

[5] I.J. Cox and M.L. Miller, "A Review of Watermarking and the Importance of Perceptual Modeling", *SPIE Conf. on Human Vision and Electronic Imaging II*, 3016(1997), pp.92-99.

[14] 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 [16]. Simple parity-based schemes abound, e.g., based on mirroring of cells, parity of row indices to which cells are assigned [16], 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.

[6] I.J.Cox, J. Kilian, F.T. Leighton and T. Shamoon, "Secure Spread Spectrum Watermarking for Multimedia", *Transactions on Image Processing*, 6(12) (1997), pp.1673-87.

[7] M. Davis and H. Putnam, "A Computing Procedure for Quantification Theory", *Journal of the ACM*, 7(3) (1960), pp. 201-215.

[8] S. Devadas, "Optimal Layout Via Boolean Satisfiability", *IEEE International Conference on Computer-Aided Design*, 1989, pp. 294-7.

[9] W.Diffie and M. Hellman, "New Directions in Cryptography", *IEEE Transactions on Information Theory*, IT-22(6), 1976, pp. 644-654.

[10] M. E. Garey and D. S. Johnson, *Computers and Intractability: a Guide to the Theory of NP-Completeness*, Freeman, 1979.

[11] J. Giraldi and M.L. Bushnell, "Search State Equivalence for Redundancy Identification and Test Generation", *Proc. Intl. Test Conf.*, 1991, pp. 184-193.

[12] M. X. Goemans and D. P. Williamson, "Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming", *Journal of the ACM*, 42(6) (1995), pp.1115-45.

[13] A.H. Hashemi, D.R. Kaeli and B. Calder, "Efficient Procedure Mapping Using Cache Line Coloring", SIGPLAN Notices, 32(5) (1997), pp.171-182.

[14] I. Hong and M. Potkonjak, "Behavioral Synthesis Techniques for Intellectual Property Protection", unpublished manuscript, 1997.

[15] L. Honey, A.H. Tewfik and K.N. Hamdy, "Digital Watermarks for Audio Signals", *Proc. of the International Conference on Multimedia Computing and Systems*, 1998, pp. 473-480.

[16] A. B. Kahng, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang and G. Wolfe, "Robust IP Watermarking Methodologies for Physical Design", *Proc. ACM/IEEE Design Automation Conf.*, 1998.

[17] D. Kirovski and M. Potkonjak, "System-Level Synthesis of Low-Power Hard Real-Time Systems", *Proc. ACM/IEEE Design Automation Conference*, 1997, pp. 697-702.

[18] J. Lach, W. H. Mangione-Smith and M. Potkonjak, "Fingerprinting Digital Circuits on Programmable Hardware", *Proc. Workshop on Information Hiding*, 1998.

[19] J. Lach, W. H. Mangione-Smith and M. Potkonjak, "FPGA Fingerprinting Techniques for Protecting Intellectual Property", *Proc. CICC*, 1998.

[20] T. Larrabee, "Test Pattern Generation Using Boolean Satisfiability", *Proc. International Test Conf.*, 11(1) (1992), pp.4-15.

[21] S.H. Low, N.F. Maxemchuk, J.T. Brassil and L. O'Gorman, "Document Marking and Identification Using both Line and Word Shifting", *Proc. INFOCOM*, 1995, pp.853-60.

[22] A.J. Menezes, P.C. van Oorschot and S.A. Vanstone, *Handbook of Applied Cryptography*, Boca Raton, CRC Press, 1997.

[23] Pretty Good (tm) Privacy, Phil Zimmerman, Users Guide: Vol. 1, Vol. 2, and Internal Data Structures from Phil's Pretty Good Software, copy: `http://www-atp.llnl.gov/atp/papers/HRM/references/pgp-refs.html`

[24] R.L.Rivest, "RFC 1321: the MD5 Message-Digest Algorithm", *Internet Activities Board*, April 1992.

[25] B. Selman, "Stochastic Search and Phase Transitions: AI Meets Physics", *IJCAI*, 1 (1995), pp.998-1002.

[26] Semiconductor Industry Association, *National Technology Roadmap for Semiconductors*, revised November 1997.

[27] D.R. Stinson, *Cryptography: Theory and Practice*, Boca Raton, CRC Press, 1995.

[28] S. Trimberger, *personal communication*, 1997.

[29] VSI Alliance, "Fall Worldwide Member Meeting: a Year of Achievement", Santa Clara, CA, October 1997.

[30] H. Wang, T. Sun, Q. Yang, "Minimizing Area Cost of on-Chip Cache Memories by Caching Address Tags", *IEEE Transactions on Computers*, 46(11) (1997), pp. 1187-1201.

[31] Xilinx, *The Programmable Logic Data Book*, Xilinx Corporation, San Jose, 1996.

[32] `http://www.rapid.org`

[33] `http://www.vsi.org`

[34] `http://www.si2.org`

[35] `http://dcs.ex.ac.uk/~aba/rsa/rc4.html`