Behavioral Synthesis Techniques for Intellectual Property Protection

FARINAZ KOUSHANFAR
University of California, Berkeley
INKI HONG
Synopsys Inc.
and
MIODRAG POTKONJAK
University of California, Los Angeles

We introduce dynamic watermarking techniques for protecting the value of intellectual property of CAD and compilation tools and reusable design components. The essence of the new approach is the addition of a set of design and timing constraints which encodes the author's signature. The constraints are selected in such a way that they result in a minimal hardware overhead while embedding a unique signature that is difficult to remove and forge. Techniques are applicable in conjunction with an arbitrary behavioral synthesis task such as scheduling, assignment, allocation, transformation, and template matching.

On a large set of design examples, studies indicate the effectiveness of the new approach that results in signature data that is highly resilient, difficult to detect and remove, and yet is easy to verify and can be embedded in designs with very low hardware overhead. For example, the probability that the same design with the embedded signature is obtained by any other designers by themselves is less than 1 in 10^{102}, and no register overhead was incurred. The probability of tampering, the probability that part of the embedded signature can be removed by random attempts, is shown to be extremely low, and the watermark is additionally protected from such tampering with error-correcting codes.

Categories and Subject Descriptors: B.5 [**Register-Transfer-Level Implementation**]; B.7.2 [**Integrated Circuits**]: Design Aids; K.5.1 [**Legal Aspects of Computing**]: Hardware and Software Protection—*Proprietary rights*

General Terms: Algorithms, Design, Security

Additional Key Words and Phrases: Intellectual property protection, watermarking, behavioral synthesis

Authors' addresses: F. Koushanfar, EECS Department, University of California, Berkeley, CA 94720; I. Hong, Synopsys Inc., Mountain View, CA 94043; M. Potkonjak, CS Department, University of California, 3532G Boelter Hall, Los Angeles, CA 90095-1596; email: miodrag@cs.ucla.edu.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 1515 Broadway, New York, NY 10036 USA, fax: +1 (212) 869-0481, or permissions@acm.org.
© 2005 ACM 1084-4309/05/0700-0523 $5.00

1. INTRODUCTION

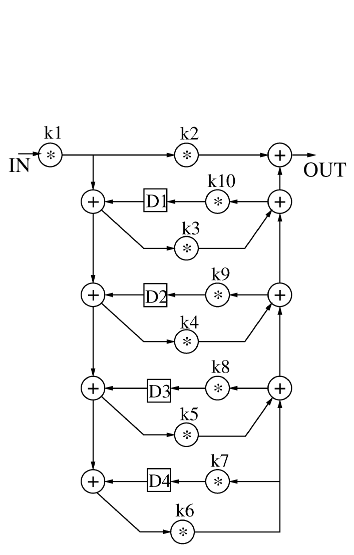
Reusable cores recently emerged as one of the most visible and important components on which future design approaches will be based. Exponential growth of both applications and implementation technology have outpaced the design productivity of the traditional synthesis process. There is a wide consensus that only through reuse of cores that the demands of pending applications and the potential ultra large-scale integration will be matched.

There is another wide consensus that viability of the new technology directly depends on the development of sound mechanisms for intellectual property protection [Girczyc and Carlson 1993; Virtual Socket Initiative[1]]. Although there are several forms of protection available for intellectual property in the electronic design automation industry that include patents, copyrights, trademarks, and trade secrets [Fernandez 1994], all these methods are insufficient and/or inapplicable for intellectual property protection of reusable cores. On the other hand, in the last few years, the rapid growth in watermarking or signature-hiding techniques in numerous areas, most often in connection with the Internet, has been observed. Numerous signature-hiding techniques have been proposed for image [Bender et al. 1996; Cox et al. 1996; van Schyndel et al. 1994], video [Hartung and Kutter 1999], voice [Boney et al. 1996] and text data [Berghel and O'Gorman 1996]. However, these techniques do not provide an adequate solution for intellectual property protection. Most of these techniques alter a large number of components in the object [Bender et al. 1996]. While in multimedia this alteration is invisible to human eyes and ears, in the case of designs and CAD tools, it will have unacceptable impact on the functionality and the correctness. A few static conventional watermarking techniques introduce relatively low distortions and may be applicable to the lowest levels of the design process such as mask production. However, these watermarks are relatively easy to detect and remove using the proper filtering process.

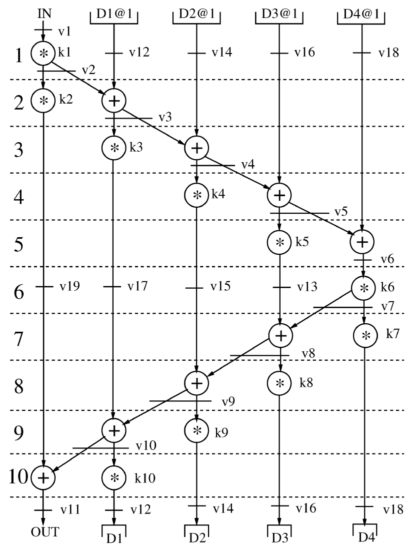
We propose a new steganography approach which is specifically developed for intellectual property protection of designs and CAD tools. The approach embeds distributed encoded messages in structural properties of designs at all levels of the design process with minimal hardware overhead while maintaining the correctness and functionality of designs. CAD tools are protected in the same way by embedding signatures in designs that they produce. The new approach is based on the key property of design-space exploration in behavioral synthesis that there are numerous competitive solutions. The idea is to select one of the competitive solutions which encodes a signature. Searching for such a solution with the embedded signature is, of course, a difficult, if not an impossible task. However, generating such a design that satisfies the original specified constraints as well as additional watermarking constraints is both easy and time efficient.

To illustrate the key ideas behind the new approach, we consider the design shown in Figure 1. The design is a 4th-order continued-fraction infinite impulse response (CF IIR) filter [Crochiere and Oppenheim 1975]. Figure 1(a)

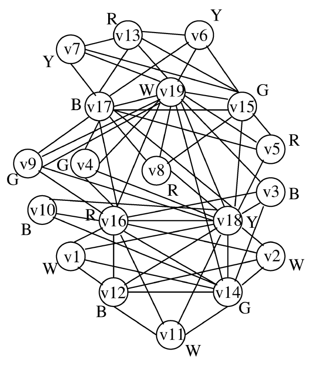
[1]Available at http://www.vsi.org.



(a) Control dataflow graph

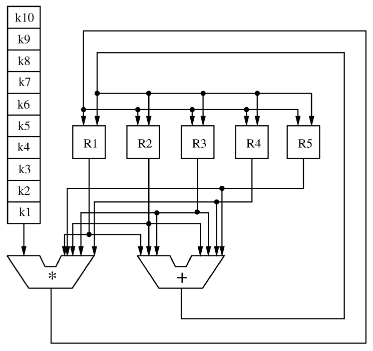


(b) Scheduled CDFG



Control Step	R1	R2	R3	R4	R5
1	v1	v12	v14	v16	v18
2	v2	v3	v14	v16	v18
3	v19	v17	v14	v16	v18
4	v19	v17	v4	v16	v18
5	v19	v17	v15	v5	v18
6	v19	v17	v15	v13	v6
7	v19	v17	v15	v13	v7
8	v19	v17	v9	v8	v18
9	v19	v10	v14	v16	v18
10	v11	v12	v14	v16	v18

Control



Datapath

(c) Colored interval graph for the scheduled CDFG

(d) Control and Datapath embedded with a signature A7

Fig. 1. Motivational example: watermarking 4th-order CF IIR filter.

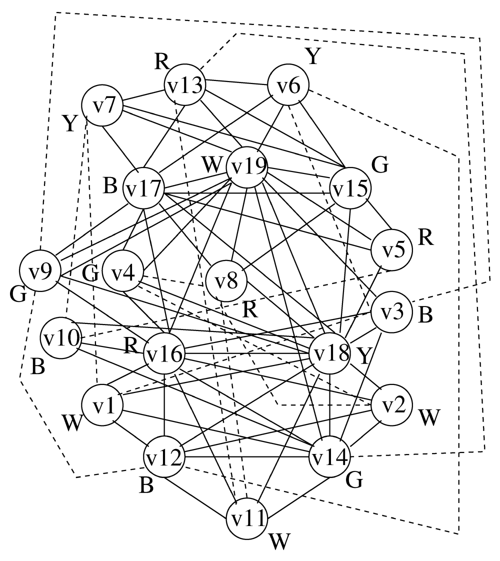
shows the control dataflow graph (CDFG) for the filter. Figure 1(b) presents a scheduled CDFG for the filter. The schedule uses 10 control steps. Values that are generated in one control step and used in a later step must be stored in a register during the intermediate control step transitions. A variable is *alive* between the time it is generated (written) and the last use (read) of it. This interval is called the *lifetime* of the variable. Two variables whose lifetimes do not overlap can be stored in the same register. The interval graph [De Micheli 1994] can be constructed as follows. For each variable, a node is made in the interval graph. Two nodes are connected if the lifetimes of the corresponding variables overlap. Register allocation can be performed by coloring the interval graph. The graph coloring problem is formally defined in the following way:

Problem: GRAPH K-COLORABILITY
Instance: Graph $G = (V, E)$, positive integer $K \leq |V|$.
Question: Is G K-colorable, that is, does there exist a function $f : V \rightarrow \{1, 2, \ldots, K\}$ such that $f(u) \neq f(v)$ whenever $\{u, v\} \in E$?

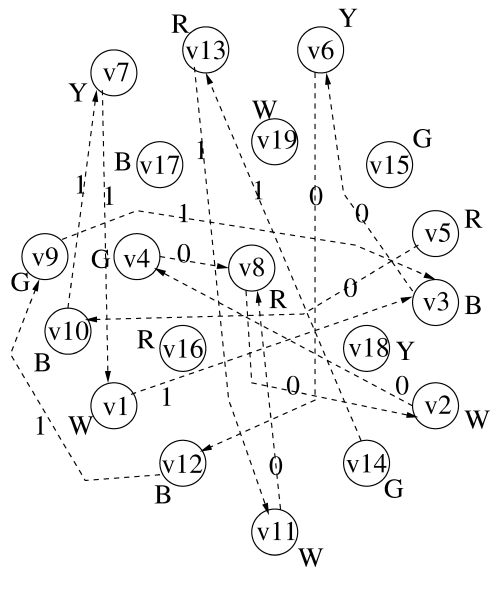
The interval graph for the example is shown in Figure 1(c). Since there are several cliques of size 5 (for example $v_1, v_{12}, v_{14}, v_{16}, v_{18}$), we need at least 5 colors to color the graph. A coloring solution with 5 colors (**W**hite, **B**lack, **G**reen, **R**ed, and **Y**ellow) is shown in Figure 1(c). Figure 1(d) shows the resulting datapath and control with 1 adder and 1 multiplier after register allocation and binding are performed. In addition to the optimality in terms of the number of registers, the control and datapath is embedded with a watermark, A7, which is extremely difficult to detect without knowing the encoding rules. The essence of the new approach is the addition of a set of extra constraints which encodes the author's signature. For the register allocation problem, the extra constraints are imposed such that a set of variables are forced to be stored in different registers which result in extra edges in the interval graph. The interval graph along with the extra edges is shown in Figure 2(a).

We assume 7-bit ASCII encoding for the characters to be used in watermarking. All the nodes in the interval graph are sorted and numbered in the increasing order of the length of their lifetimes. In the increasing order of node numbers, each node is considered for embedding a single bit. After all the nodes are considered, we repeat the process from the first node in the order. For each bit in the watermark, the terminal node is chosen such that the terminal node number represents the bit value. To embed 1, the terminal node of the added edge should have an odd number. To embed 0, the terminal node of the added edge should have an even number. The first feasible node in the increasing order of node numbers, starting from the position right after the terminal node of the last edge added, is selected as the terminal node to embed the bit value. The added edges with their encoded values for our example are shown in Figure 2(b). They encode 10000010110111 in binary which is equal to A7 in the ASCII code. For graphs with reasonably many nodes, the probability that any register allocation method will result in the same solution with the watermark is extremely low. For example, the interval graph in Figure 1(c) has 35 unique coloring solutions with 5 colors. Therefore, there is a 2.9% chance that any good register allocation method finds the same solution, if we assume uniform probability for all possible 5-color solutions. Note that the proposed approach targets, and is progressively more effective for, large designs.

We introduce a dynamic watermarking technique for protecting the value of intellectual property of CAD tools and reusable core components. The essence of the new approach is the addition of a set of design and timing constraints as a preprocessing design step which encodes the signature of the author. The technique is dynamic: the watermarking constraints are chosen as a preprocessing step before the synthesis is performed and thus influence the final design. The constraints are selected in such a way that they result in the minimal overhead



(a) Colored interval graph with the extra edges added



(b) Extra edges with encoded values

Fig. 2. Illustrated explanation of how the signature A7 is embedded in the register allocation solution shown in Figure 1(c).

in area, throughput, testability, and/or power consumption, while embedding the signature which is unique and difficult to detect and remove. The idea is based on the key property of design-space exploration in behavioral synthesis that consists of several interdependent NP-complete subsynthesis tasks where there exist many competitive solutions with similar quality.

The proposed technique has a spectrum of important unique properties. The technique is transparent to manual and automatic design processes and therefore can be used in conjunction with any available or future set of design tools. The generic technique is applicable to all design steps. It also provides the designers with the ability to easily embed their signatures in many different ways. Note that, in addition to the approaches discussed in this article, in a more comprehensive technical report [Koushanfar et al. 2003], we show how this scheme can be applied to several other behavioral synthesis tasks such as partitioning, scheduling, transformations, and template matching.

The introduction of the encoded quantitative message in the design enables the connection to the use of cryptographic techniques which, in turn, provide many exciting options for protection, including remotely erasable signatures. The technique allows many degrees of freedom to the designer regarding the selection of signatures such as which encoding schemes should be used, how many different signatures should be used, how large they should be, and exactly in which parts of the design they should be embedded.

From the point of view of watermarking technique, another novelty is the connection of watermarking to error-correcting codes which opens numerous possibilities for exponentially improving the reliability of the generic intellectual property protection techniques and provides a mechanism for detection of attempts for signature removal by unauthorized persons. We believe that simplicity, strength, low hardware overhead, distributive nature, resilience against tampering, and complete transparency will make the proposed technique the method of choice for industrial strength protection of designs, CAD tools, and algorithmic intellectual property protection.

2. RELATED WORK

In this Section, we review the most relevant related work on intellectual property protection, watermarking for multimedia and functional artifacts, and cryptographic techniques. The cryptographic techniques are used either to supplement the watermarking methods or in conjunction with them.

Data watermarking, also known as data hiding, a form of steganography [Johnson et al. 2001; Katzenbeisser and Petitcolas 2000], embeds data into digital media for the purpose of identification, annotation, and copyright. Throughout history, a multitude of methods and variations have been used to hide information. Recently, the proliferation of digitized media and the Internet revolution have been creating a pressing need for copyright enforcement schemes to protect copyright ownership. We can classify watermarking research along the following lines: watermarking of nonfunctional artifacts, watermarking of functional artifacts, other intellectual property protection techniques, and cryptographic attacks.

Nonfunctional artifacts themselves can be classified into two categories: computer generated and from other sources. The latter refers to watermarking of digital texts, images, audio, and video data. The former group consist of animation (cartoon), maps, and compact representation of two- or three-dimensional objects.

The main idea in watermarking computer-generated artifacts is to conduct interpolation of the lines and planes dictated by the signature in such a way that the maximal discrepancy between the original specification and the watermarked instance is maintained within the user specified limits. Several techniques have been proposed, for example, [Yeo and Yeung 1999], Benedens and Busch [2000], and Ohbuchi et al. [2000] introduced watermarking schemes for two- and three-dimensional graphical objects. Khanna and Zane [2000] conduct watermarking on map objects. Benedens and Busch [2000] present a watermarking scheme for computer animation (meshes).

For texts, three methods have been proposed for applying watermarking: text-line coding, word-space coding, and character coding [Berghel and O'Gorman 1996]. Brassil et al. [1999] present a summary of the copyright protection for the electronic distribution of text documents. Bender et al. [1996] evaluate various techniques for data hiding in digital media in light of three applications: copyright protection, tamper proofing, and augmentation data embedding. Craver et al. [1996] discussed a possible attack to any watermarking scheme for multimedia images and proposed a method which is resilient to such attack. Two methods for watermarking images which are highly sensitive to noise and are easily corrupted are proposed in Schyndel et al. [1994]. A promising method based on spread-spectrum which is difficult to intercept and remove but may introduces perceivable distortion into the host signal has been proposed by Cox et al. [1996]. Yeung et al. [1997] addressed the requirements, techniques, and applications of digital watermarking targeted for high-quality images. An overview of image watermarking can be found in Hartung and Kutter [1999]. Kutter and Winkler [2002] propose a perceptual model for hiding a spread-spectrum watermark of variable amplitude and density in an image. Podilchuk and Zeng [1997] proposed a method to apply visual models developed for image compression to digital watermarking of images which resulted in transparent and robust watermarking. Hartung and Kutter [1999] presented a scheme for robust interoperable watermarking of MPEG-2 encoded video in video broadcast applications. Swanson et al. [1997] presented a watermarking procedure for video where individual watermarks were created for objects within the video. Boney et al. [1996] presented a technique for embedding digital watermarks into audio signals. A novel audio watermarking scheme based on spread-spectrum was developed by Kirovski and Malvar [2001].

In watermarking functional artifacts, several methods have been proposed since the original submission of this article. The watermarking-based hardware IPP technique can be classified according to two criteria: the level of abstraction used for augmentation of the design specification with signature and according to employed watermarking protocol. Watermarking techniques have been proposed at all level of design process, including the algorithm-level [Kim et al. 1998; Rashid et al. 1999; Chapman and Durrani 2000], high-level [Kirovski et al. 1998; Kirovski and Potkonjak 2003], logic synthesis [Meguerdichian and Potkonjak 2000; Oliveira 2001], FPGA-based physical level synthesis [Lach et al. 1998, 2001], and physical synthesis [Charbon 1998; Torunoglu and Charbon 1999; Kahng et al. 2001; Newbould et al. 2002]. Also, in addition to watermarking digital designs, several efforts were dedicated to watermark analog and mixed signal designs [Newbould et al. 2001; Irby et al. 2001].

In addition to being the first watermarking-based IPP scheme at the time of the initial submission of the first version of this article, there are several distinctions between this work and other publications. First, it is important to observe that it is most beneficial to apply watermarking as early as possible in the design process. This is so because, in this situation, all the consequent tasks are also protected. In addition, there are many steps during behavioral synthesis, instances of the problem have a large number of components, and the induced overhead has a good chance of being low because it is difficult to predict the final impact of the proposed architectural solution. Also, almost all watermarking techniques proposed in this article are essentially directly applicable to software protection, the market segment where piracy is a much larger problem than in hardware protection. Finally, this is only article that discusses the potential attacks on the IPP hardware schemes.

There are a number of other relevant intellectual property protection and detection techniques that are also the subject of active research. Obfuscation is an approach that takes a program (or a circuit) specification as its input P and produces a new description F(P) that has the same functionality as P, which is “unintelligible” [Hada 2000]. Hardware and software metering is a technique that enables reliable low-overhead proofs for the number of manufactured parts and copied programs [Koushanfar et al. 2001]. SiidTech has proposed an approach for integrated circuit identification from random threshold mismatches in an array of addressable MOSFETs. The technique leverages on the random threshold mismatches formed during fabrication [Lofstrom et al. 2000]. Other related research areas are reverse engineering and forensic engineering. Reverse engineering is a method to extract the original design based on the attributes of a few instances of the designed data, hardware, or software [Kumagai 2000]. Computational forensic engineering (CFE) identifies the entity that created a particular intellectual property (IP). Rather than relying on watermarking content or design, the generic CFE methodology analyzes the statistics of certain features of a given IP and quantities the likelihood that a well-known source has created it. Recently, a computational forensic engineering technique for IPP have been proposed by Wong et al. [2001]. Another related research area that is worth mentioning is security of the C code. Wagner et al. [2000] presents a statistical analysis scheme for finding potential buffer overrun vulnerabilities in security-critical C code. Secure Internet delivery of the integrated circuit from the design house to the fabrication company is yet another related active area of research.

One approach to data security is to use cryptographic techniques. The cryptographic techniques can be used to supplement the watermarking methods, for example, by encrypting the embedded watermark. In cryptology, the information is scrambled using an encryption transformation before it is sent, and the information can be viewed after de-scrambling with the decryption transformation. Note that cryptography does not put the signature in the information, but restricts access to the document. A public-key cryptosystem can be used to implement an electronic mail system in which messages are kept private and can be signed [Schneier 1996].

3. TECHNICAL BACKGROUND

We selected as our computational model synchronous dataflow (SDF) [Lee and Messerschmitt 1987]. The SDF is a special case of dataflow in which the number of data samples produced or consumed by each node on each invocation is specified a priori. Nodes can be scheduled statically at compile time onto programmable processors. We restrict our attention to homogeneous SDF (HSDF) where each node consumes and produces exactly one sample on every execution. The HSDF model is well suited for specification of single task computations in numerous application domains such as digital signal processing, video and image processing, communications, control, information and coding theory, and multimedia.

The syntax of a targeted computation is defined as a hierarchical control-dataflow graph (CDFG) [Rabaey et al. 1991]. The CDFG represents the computation as a flow graph with nodes, data edges, and control edges. The semantics underlying the syntax of the CDFG format, as we already stated, is that of the synchronous dataflow model.

The proposed method can be also applied to other computational and hardware models. Note that since the proposed method is based on adding more constraints, it is likely that it will be more efficient in more complex computational and hardware models. However, it is important to note that, due to the much more dynamic nature of the code, it may be significantly harder to perform signature detection. Therefore, additional studies are required to asses the viability of the proposed watermarking schemes in other domains.

Behavioral synthesis transforms a given behavioral specification into a register-transfer level (RTL) description that can implement the given behavior. Many research works in behavioral synthesis have been performed in the past decade. Behavioral synthesis is now considered a mainstream CAD activity and several products are commercially being offered. Behavioral synthesis encompasses a variety of synthesis tasks such as scheduling, allocation, binding, partitioning, module selection, and transformations. An overview of the field can be found in De Micheli [1994].

Cryptographic techniques are relevant to the first step of the proposed intellectual property protection techniques. The author's signature data are enciphered and then embedded. The specific method we use involves a cryptographic hash function such as MD5, a public key cryptosystem such as RSA, and a stream cipher such as RC4 [Schneier 1996].

A cryptographic hash function is a collision-free one-way hash function H that takes variable-length input x and returns a fixed-length output $H(x)$. The collision-free property means that finding messages x and y such that $H(x) = H(y)$ is computationally intractable. The one-way property means that given a hash value h, finding an input x such that $H(x) = h$ is computationally intractable.

A public key cryptosystem is a cryptosystem that allows one to safely publish his/her key required to encrypt a message sent to the person. One has a private key that allows him/her to decrypt the messages encrypted with his/her public key. The decrypting key should not be easily computed from the encrypting key. The security of RSA is based on the difficulty of computing prime factorization of large numbers. A stream cipher is a generator of a pseudo-random bitstream called *keystream*. This *keystream* is combined with a plaintext message by a bitwise exclusive-or operation to produce a ciphertext.

To improve the reliability of the proposed intellectual property protection techniques, an error-correcting and detection code for the embedded signature can be employed. The error-correcting and detection code can provide a mechanism for detection of attempts for signature removal by unauthorized persons. In the implementation, the BCH codes [Lin and Costello 1983] have been used to encode the signature data. For any positive l ($l \geq 3$) and m ($m < 2^{l-1}$), there exists binary m-bit-error-correcting BCH codes for the k-bit signature, where $k = 2^l - 1$ [Lin and Costello 1983].

4. OBJECTIVES AND METRICS FOR WATERMARKING

In this section, we lay out the technical foundations required to develop a consistent quantitative approach for intellectual property protection using watermarking techniques. While the data-hiding techniques are to a certain extent inevitably related to subjective criteria about what authorized users, potential nonauthorized users, and sellers are able to detect and remove, we focus our attention on the most relevant objective criteria. We first analyze the key watermarking properties which imply the quality of the associated data-hiding technique. We conclude the section by summarizing the selected numerical metrics which quantify the effectiveness of a proposed steganography technique for design properties.

4.1 Objectives

In order to be effective, the watermark should exhibit the following properties.

—*Correctness of Functionality.* The correctness of functionality such as timing and design requirements should not be affected by the watermark.

—*Low Hardware Overhead.* The watermark should result in low design performance (e.g. area, testability, and power consumption) overhead.

—*Transparency.* The addition of a watermark to designs by CAD tools should be completely transparent so that it can be used for any existing CAD tools and designs. Therefore, watermarking should be done either as preprocessing or postprocessing step.

—*Proof of Authorship.* The watermark should be readily detectable by the owner and law enforcement authorities. The watermark should unambiguously identify the owner and should be accompanied with the convincing mathematical proof on its quality.

—*Difficult to Locate.* The watermark should be unobtrusive (perceptually invisible). If thieves could find it, they would solve the most difficult piece of the watermark removal problem. Note that being difficult to detect implies that special knowledge is required to detect the watermark.

—*Resilience.* The watermark must be difficult or impossible to remove, at least without significantly degrading the quality of the original design. It must be difficult to remove by any techniques which do not have complete knowledge of the design. In particular, the resilience against the standard optimization techniques and local reengineering approaches is important. Furthermore, the watermark should be fault tolerant so that even if part of it is removed, the authorship message is still preserved.

—*Proportional Part Protection.* The watermark should be distributed all over the design in order to facilitate the protection of not only the complete design, but also its parts. Ideally, more valuable parts are proportionally more strongly protected.

—*Signature Detection.* The watermark should be easy to detect. In addition, the cost of detecting the presence of the signature should be low. Finally, it is preferable that the signature detection and demonstration can be accomplished in such a way that its exact position and structure are not revealed.

Note that these objectives are often mutually conflicting. For example, making the watermark large and more difficult to remove is likely to result in more degradation in design metrics.

4.2 Metrics

From the objectives for the watermark, the following metrics which allow the comparison and evaluation of different watermarking techniques can be identified.

—*Strength of the Authorship Proof.* The probability of *coincidence*, P_C, that the same design with the watermark is produced by any other authors must be minimized. The probability is proportional to the probability that any specific design is produced by a synthesis tool or by a manual design. For example, if there exist k designs of the same quality, we assume that the probability that any one of them is produced by a design procedure is $1/k$.

—*Resilience.* The better the watermark, the more difficult it is to be removed. There are several objectives that indirectly allow low removability. First, the invisible watermark can prevent some thieves from immediately attempting to remove it. If thieves could find it, they would try to remove it. Second, providing more fault tolerance to the watermark can be instrumental in achieving a lower level of removability. Third, when the watermark is distributed all over the design, it is more likely to prevent its removal. Therefore, we numerically define the resilience by the following parameters: (i) the probability that k bits of the watermark is removed by random tampering, for example, changing a register assignment of one variable in register binding, (ii) the number of bits in the watermark, and (iii) the percentage of the number of bits which can be removed while preserving the initially encoded authorship message.

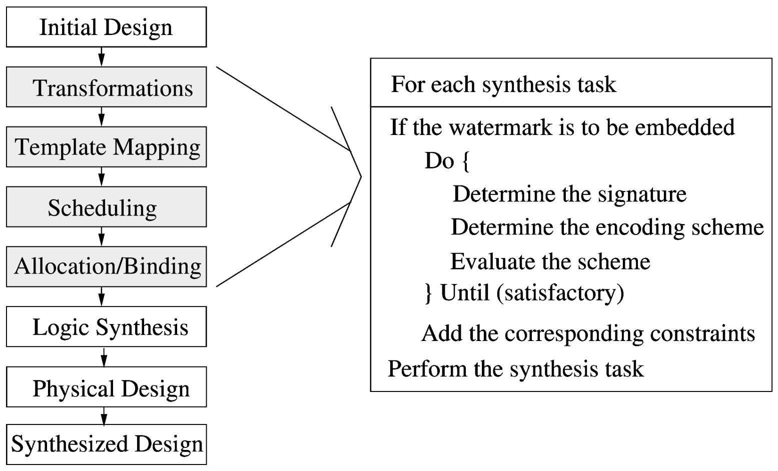


Fig. 3. The global flow of the generic approach.

—*Design Metrics Degradation.* The degradation of the design metrics by the watermark must be minimized. This metric is conflicting with the *resilience* metric. Therefore, the levels of resilience and design metrics degradation must be carefully determined. The design metrics degradation is quantitatively expressed by a percentage alteration of the relevant design metrics due to the embedding of the watermark.

5. INTELLECTUAL PROPERTY PROTECTION USING WATERMARKING

The generic approach for watermarking designs is shown in Figure 3. The essence of the new approach is the addition of a set of design and timing constraints which encodes the author's signature. The constraints are selected in such a way that they result in minimal hardware overhead impact, while embedding the signature to make it difficult to detect and remove. The selection of the constraints depends on the employed encoding scheme.

Distributors of intellectual property such as hardware design, software, documents, and images wish to prevent registered users from releasing unauthorized copies. Finger-printing allows a distributor to detect any unauthorized copy and trace it back to the user by uniquely marking and registering each copy of the intellectual property.

6. WATERMARKING FOR REGISTER ALLOCATION AND BINDING

6.1 Signature Data Selection, Encoding, Encryption, and Protection

The two variables which do not have overlapped lifetimes can share a register. The interval graph describes the register-sharing possibility between variables. Register allocation is performed by coloring the interval graph using the graph-coloring algorithm. Therefore, as shown in the motivational example in Section 1, the watermark for register allocation and binding solutions can be embedded in designs. First, the signature data to be embedded should be selected. The signature data should be selected such that they uniquely identify the author as the owner of the design.

The selected signature data need to be encoded so that they can be embedded in the design. We note that different encoding schemes will result in different quality watermarking solutions. However, by using a cryptographic technique, the encoded message, by any encoding scheme, can be transformed to a pseudo-random bitstream. After selecting the encoding scheme for the signature data, the author's encoded signature data are enciphered and then embedded. This step results in two advantages. First, it strengthens the proof of authorship by allowing only the designer with the secret key to decipher the signature data. Next, it makes the signature data look like a random bitstream so that the detection of the signature data by unauthorized users, using any type of statistical analysis, becomes harder.

The message is enciphered in the following steps. First, the author's signature data is applied to MD5 cryptographic hash function. The generated hash is encrypted by the public key of the designer using RSA public key cryptosystem. Using the obtained cipher as an input, RC4 stream cipher generates a pseudo-random keystream. This keystream is combined with the plaintext signature data by a bitwise exclusive-or operation to produce the ciphertext signature data. Finally, the ciphertext signature data are embedded as extra constraints in the design.

As described in the motivational example in Section 1, we assume the ASCII encoding for the characters to be used in watermarking for graph coloring. All the nodes in the interval graph are sorted and numbered in the increasing order of the length of their lifetimes. In the increasing order of node numbers, each node is considered for embedding a single bit. Each consecutive 7-added edges represent a character in the ASCII code. After all the nodes are considered, we repeat the process from the first node in the order. For each bit in the watermark, the terminal node is chosen such that the terminal node number represents the bit value. To embed 1, the terminal node of the added edge should be an odd number. To embed 0, the terminal node of the added edge should be an even number. The first feasible node in the increasing order of node numbers, starting from the position right after the terminal node of the last edge added, is selected as the terminal node to embed the bit value. This scheme is to add extra edges more evenly in the graph.

To improve the reliability of the proposed intellectual property protection techniques against tampering of design, an error-correcting and detection code for the embedded signature is employed. The m-bit error-correcting BCH codes [Lin and Costello 1983] have been used to encode the signature data.

6.2 Signature Detection

No watermarking scheme is complete without a signature detection procedure. The goal of the signature detection procedure is to check the existence of an alleged integrated circuit-specific watermark. Signature detection can be accomplished in a variety of ways. Different approaches provide different trade-offs in terms of cost, time effort, and preservation of the integrated circuits. All signature detection techniques themselves have two phases. The first phase is collecting the information about the relevant structural properties of a design and complete or partial reconstruction of the design specification at one or more levels of abstraction. The second phase is signature verification. The goal of this phase is to evaluate the presence of the constraints that correspond to the signature in the design under inspection. This phase can be easily accomplished using an inspection and comparison with the initial watermarked design. Therefore, the second phase is identical for signature detection procedures and can be done in linear time with respect to the number of superimposed constraints. Note that zero knowledge techniques can be used to demonstrate the presence of the signature without revealing its location and content [Schneier 1996]. There are a number of alternatives for the first phase, including the following.

Reverse engineering. There are a number of research and industrial labs that have developed a variety of technologies for reverse engineering. Complex state-of-the-art designs can be reverse engineered in a few days. Usually, the protective layers of integrated circuits are removed, and the physical layout is scanned. After that, this information is processed, reverse engineered, and the specifications at higher levels of abstraction are obtained. Several references provide additional information about reverse engineering process for integrated circuits [Anderson and Kuhn 1996; Kumagai 2000].

Measuring time delays of gate-level signals and energy consumption. Recently, a number of very popular power and time delay measurement-based techniques have been proposed as an effective means for breaking the security of integrated circuits that realize cryptographical protocols [Kocher 1996, 1999]. These techniques enable the observer to draw conclusions about the operations conducted in a unit by observing the energy consumption or the time-of-arrival of a signal on a particular port.

Fault-inducing radiation. Another effective way for attacking cryptographical protocols is based on exploring the faults induced after a design is exposed to radiation. It has been demonstrated that after the faults are induced, one can deduce enough about the internal structure of the design and secret values to break popular cryptographical protocols. Similarly, the observed information can be used to obtain the details of how a particular functionality is implemented and, therefore, can facilitate the signature detection process.

Access using don't care states. Finite state machines (FSMs) of many designs typically have one or more don't care states. These states can be used to initialize sequences for traversing all of the states in FSM. Once one has the knowledge about the FSMs that control reads and writes to the registers, reconstruction of the coloring of the conflict graph is an easy task. Note that this technique can be further augmented using test-scan and debugging hardware resources.

Numerical stability-based methods. Several behavioral synthesis steps alter numerical stability of the design. If these steps are properly conducted, the design still provides solutions that are within user specified levels of tolerance.

The final remark regarding signature detection is that its detection is not impacted by the design modifications at the logic synthesis or the physical design level (e.g., changes of layout) because these modifications do not alter design specifications at the behavioral level. On the other hand, changes at the behavioral level imply a very high probability for changes at lower levels and provide additional protection against these alternations. Note that after minor changes at the behavioral level, one can use the modified copy detection techniques to identify parts of the watermark in nonaltered paths.

6.3 Statistical Calculation of Intellectual Property Protection Strength

The probabilities of coincidence and tampering for the watermark are especially primary indicators of its protection strength. We note that the use of the terminology "probability" does not follow its exact meaning from mathematics in a rigorous sense. Rather, the term "probability" in this subsection is used as an approximation to the actual probability. We provide the formula to compute the probabilities for random graphs in graph coloring. For other synthesis tasks, the probabilities can be defined and computed similarly. For a random graph $G_{n,p}$ with n nodes and edge probability p, suppose c colors are used, and k edges are added as watermarking constraints. Let P_C denote the probability of generating the same coloring solution with the signature, given that the solution uses the same number of colors.

$$P_C = \left(1 - \frac{1}{c}\right)^k.$$

The probability that any register allocation method will result in the same solution as the watermark is extremely low for graphs with reasonably many nodes. For example, consider a random graph $G_{1000,0.1}$ with 23 colors. Suppose the watermark uses 4990 extra edges. The probability that any two nodes with no connecting edges are assigned different colors is $1 - \frac{1}{23}$. The probability that all 4990 pairs of nodes in the watermark are assigned different colors is $(1 - \frac{1}{23})^{4990} \approx 4.65 \times 10^{-97}$. Experimental results have shown that no register overhead for this watermark is incurred. Let P_T denote the probability of removing one or more signature bits by changing the colors of one node. We approximate the P_T by

$$P_T = 1 - \left(1 - \frac{k}{\frac{n(n-1)(1-p)}{2}}\right)^{\frac{n}{c}-1}.$$

This is so because, on average, there are n/c nodes with the same color. The probability that no signature bit is removed can be approximated by the probability that there are no encoded edges between the n/c nodes with the same color.

From these formulas, we note that P_C decreases exponentially as the number of bits in the signature increases. Thus, it is better to use a larger signature to prove the authorship. We also note, however, that P_T increases as the number of bits in the signature increases which means that it is better to use a smaller signature to prevent tampering. These two requirements are mutually contradictory. The problem can be resolved by using error-correcting codes. We can achieve low P_C and P_T by embedding a large signature with m-bit error-correcting codes where m should increase as the number of bits in the signature increases. The probability of tampering after m-bit error-correcting codes are used is approximated by $\sum_{i=m+1}^{\frac{n}{c}} P_i$, where P_i is the probability that i

Table I. Watermarking of the Random Graph Examples During Graph Coloring

# of Nodes	P(Edge)	# of Colors	# of Edges Added for k More Colors				
			$k = 0$	$k = 1$	$k = 2$	$k = 3$	$k = 4$
100	0.1	5	85	195	490	690	795
	0.3	10	100	290	385	685	790
	0.5	16	165	250	305	585	785
200	0.1	8	195	795	1185	1590	2175
	0.3	16	180	605	790	1560	2055
	0.5	26	150	755	950	1380	1815
500	0.1	14	575	2570	3855	5340	7500
	0.3	32	560	1630	3250	4955	5875
	0.5	52	1305	2095	3860	4450	6340
1000	0.1	23	4995	6595	9790	12985	14660
	0.3	55	2995	6495	7995	9980	14585
	0.5	89	850	1495	2975	4500	5930
2000	0.1	38	8855	19990	28505	39865	50010
	0.3	95	7650	18560	27635	36780	45310
	0.5	157	6560	11290	18925	24915	30265

encoded edges are removed by changing the colors of one node. P_i is approximated by $\frac{P_T}{2^i}$.

7. EXPERIMENTAL RESULTS

To compare and evaluate the effectiveness and efficiency of the heuristic algorithms, standard test cases from the random graph model are used. In the random graph model, for a graph $G_{n,p}$ with n nodes, edges are generated independently with probability p between each pair of nodes. Although unlikely to have much relevance to practical applications of graph coloring, such a testbed has the advantage, in a sense, that behavior of any heuristic on one graph of this type is a good predictor for its behavior on any other [Johnson et al. 1991].

For another research project, we developed a heuristic for the graph coloring problem which outperforms the best heuristics reported in [Johnson et al. 1991]. Extensive experimentation has shown that the algorithm outperforms the state-of-the-art techniques reported in [Johnson et al. 1991] in solution quality with an order of magnitude runtime improvement.

We used the standard random graphs as test cases for our approach to the graph coloring problem. Table I provides the experimental results for various cases. The first column presents the number of nodes in the graph and the second column the edge probability. The third column provides the number of colors used for the original random graph. The fourth to eighth columns present the number of edges that can be added before k more colors are added, where $k = 0, 1, 2, 3, 4$, respectively. When adding edges, 5 edges were added at a time, which explains the fact that all the numbers in Table I are multiples of 5. Since the purpose of this experimentation is not to achieve the minimum possible solution but to generate good quality solutions fast in many times for the demonstration of the effectiveness of our approach, we restrict the running time of the heuristic within a reasonable amount of time. For example, the running time for the graph $G_{1000,0.5}$ is limited to 10 minutes on a SUN Sparc5 workstation. Table II presents the probabilities of coincidence and tampering

Table II. P_C and P_T for the Watermarks on the Graphs in Table I

Graph	P_C/P_T for k More Colors		
	$k = 0$	$k = 1$	$k = 2$
$G_{100,0.1}$	$2 \times 10^{-8}/4 \times 10^{-2}$	$9 \times 10^{-16}/9 \times 10^{-2}$	$3 \times 10^{-33}/2 \times 10^{-1}$
$G_{100,0.3}$	$4 \times 10^{-5}/7 \times 10^{-3}$	$2 \times 10^{-12}/2 \times 10^{-2}$	$4 \times 10^{-15}/3 \times 10^{-2}$
$G_{100,0.5}$	$3 \times 10^{-5}/4 \times 10^{-3}$	$4 \times 10^{-7}/8 \times 10^{-3}$	$4 \times 10^{-8}/1 \times 10^{-2}$
$G_{200,0.1}$	$10 \times 10^{-12}/2 \times 10^{-2}$	$4 \times 10^{-41}/6 \times 10^{-2}$	$1 \times 10^{-54}/9 \times 10^{-2}$
$G_{200,0.3}$	$1 \times 10^{-5}/2 \times 10^{-3}$	$2 \times 10^{-16}/6 \times 10^{-3}$	$3 \times 10^{-20}/9 \times 10^{-3}$
$G_{200,0.5}$	$3 \times 10^{-3}/5 \times 10^{-4}$	$5 \times 10^{-13}/3 \times 10^{-3}$	$1 \times 10^{-15}/4 \times 10^{-3}$
$G_{500,0.1}$	$5 \times 10^{-19}/4 \times 10^{-3}$	$1 \times 10^{-77}/2 \times 10^{-2}$	$1 \times 10^{-108}/2 \times 10^{-2}$
$G_{500,0.3}$	$2 \times 10^{-8}/4 \times 10^{-4}$	$2 \times 10^{-22}/1 \times 10^{-3}$	$8 \times 10^{-43}/2 \times 10^{-3}$
$G_{500,0.5}$	$1 \times 10^{-11}/2 \times 10^{-4}$	$5 \times 10^{-18}/4 \times 10^{-4}$	$5 \times 10^{-32}/7 \times 10^{-4}$
$G_{1000,0.1}$	$5 \times 10^{-97}/4 \times 10^{-3}$	$2 \times 10^{-122}/6 \times 10^{-3}$	$3 \times 10^{-174}/9 \times 10^{-3}$
$G_{1000,0.3}$	$1 \times 10^{-24}/2 \times 10^{-4}$	$2 \times 10^{-51}/5 \times 10^{-4}$	$4 \times 10^{-62}/6 \times 10^{-4}$
$G_{1000,0.5}$	$7 \times 10^{-5}/2 \times 10^{-5}$	$6 \times 10^{-8}/3 \times 10^{-5}$	$6 \times 10^{-15}/6 \times 10^{-5}$
$G_{2000,0.1}$	$3 \times 10^{-103}/9 \times 10^{-4}$	$3 \times 10^{-226}/2 \times 10^{-3}$	$3 \times 10^{-314}/3 \times 10^{-3}$
$G_{2000,0.3}$	$7 \times 10^{-36}/6 \times 10^{-5}$	$4 \times 10^{-85}/1 \times 10^{-4}$	$4 \times 10^{-125}/2 \times 10^{-4}$
$G_{2000,0.5}$	$6 \times 10^{-19}/1 \times 10^{-5}$	$7 \times 10^{-32}/2 \times 10^{-5}$	$1 \times 10^{-52}/3 \times 10^{-5}$

Table III. The Overhead for $P_C = 10^{-10}$ and $P_T = 10^{-10}$ for the Graphs in Table I

Graph	# of Bits in Signature	# of Bit Errors to be Corrected	# of Parity Bits for Error-Correcting Codes	# of Extra Colors Used
$G_{100,0.1}$	127	21	98	1
$G_{100,0.3}$	255	11	84	1
$G_{100,0.5}$	511	6	54	3
$G_{200,0.1}$	255	23	156	1
$G_{200,0.3}$	511	12	108	1
$G_{200,0.5}$	1023	7	70	2
$G_{500,0.1}$	511	36	270	0
$G_{500,0.3}$	1023	16	160	1
$G_{500,0.5}$	2047	10	110	1
$G_{1000,0.1}$	1023	24	235	0
$G_{1000,0.3}$	2047	19	209	0
$G_{1000,0.5}$	4095	11	132	3
$G_{2000,0.1}$	1023	13	130	0
$G_{2000,0.3}$	1023	15	150	0
$G_{2000,0.5}$	2047	14	154	0

for the watermarks in Table I. For each case, the maximum number of edges that can be added before one more color is required is used to compute the probabilities. We note that P_T is high for almost all the cases. m-bit-error-correcting BCH codes [Lin and Costello 1983] are employed to reduce the probability. In Table III, we show the overhead for achieving $P_C = 10^{-10}$ and $P_T = 10^{-10}$ in terms of the number of parity bits used for error-correcting codes and the number of extra colors. In Tables IV and V, we also show the overhead for achieving $P_C = 10^{-20}$, $P_T = 10^{-20}$ and $P_C = 10^{-50}$, $P_T = 10^{-50}$, respectively. We also applied our approach on 15 real-life designs. The designs are the following: two GE controllers, two Honda controllers, a wavelet filter, an NEC digital-to-analog converter (DAC), two components of modems, a linear controller for automotive motion control, a LMS audio formatter, a speech compressor, a CORDIC,

Table IV. The Overhead for $P_C = 10^{-20}$ and $P_T = 10^{-20}$ for the Graphs in Table I

Graph	# of Bits in Signature	# of Bit Errors to be Corrected	# of Parity Bits for Error-Correcting Codes	# of Extra Colors Used
$G_{100,0.1}$	127	21	98	1
$G_{100,0.3}$	511	9	81	3
$G_{100,0.5}$	511	7	63	3
$G_{200,0.1}$	255	25	164	1
$G_{200,0.3}$	511	13	117	1
$G_{200,0.5}$	1023	8	80	3
$G_{500,0.1}$	511	36	270	0
$G_{500,0.3}$	1023	17	165	1
$G_{500,0.5}$	2047	11	121	1
$G_{1000,0.1}$	1023	25	245	0
$G_{1000,0.3}$	2047	20	220	0
$G_{1000,0.5}$	4095	12	144	3
$G_{2000,0.1}$	1023	17	170	0
$G_{2000,0.3}$	2047	15	165	0
$G_{2000,0.5}$	8091	13	169	1

Table V. The Overhead for $P_C = 10^{-50}$ and $P_T = 10^{-50}$ for the Graphs in Table I

Graph	# of Bits in Signature	# of Bit Errors to be Corrected	# of Parity Bits for Error-Correcting Codes	# of Extra Colors Used
$G_{100,0.1}$	255	25	164	6
$G_{100,0.3}$	511	13	117	10
$G_{100,0.5}$	1023	7	70	12
$G_{200,0.1}$	255	30	184	2
$G_{200,0.3}$	511	17	153	5
$G_{200,0.5}$	1023	16	160	8
$G_{500,0.1}$	511	38	288	1
$G_{500,0.3}$	1023	25	245	3
$G_{500,0.5}$	2047	19	209	4
$G_{1000,0.1}$	1023	25	245	0
$G_{1000,0.3}$	2047	20	220	0
$G_{1000,0.5}$	4095	25	300	7
$G_{2000,0.1}$	2047	20	220	0
$G_{2000,0.3}$	8191	15	195	1
$G_{2000,0.5}$	16383	10	140	2

2D Wang DCT, 2D Arai DCT, and 2D Lee DCT. We have chosen the small-and moderately-sized designs to demonstrate the effectiveness of the approach for most difficult examples. As the design is smaller, it is harder to embed large signatures. Table VI provides the experimental results for the designs. The second column presents the number of variables in the designs, and the third column the number of registers used for the original design. The fourth to eighth columns present the number of edges that can be added before k more registers are used, where $k = 0, 1, 2, 3, 4$, respectively. In Table VII, we show the overhead for achieving $P_C = 10^{-50}$ and $P_T = 10^{-50}$, in terms of the number of parity bits used for error-correcting codes and the number of extra colors. We assume that the edge probability in all designs is 0.3. It is a moderate assumption since most of the designs result in sparse interval graphs.

Table VI. Experimental Results of Watermarking Designs in Register Allocation

Design	# of Variables	# of Registers	# of Edges Added for k More Registers				
			$k = 0$	$k = 1$	$k = 2$	$k = 3$	$k = 4$
GE Controller 1	48	17	25	35	60	100	150
GE Controller 2	108	26	85	155	400	460	710
Honda Controller 1	97	29	60	100	190	340	690
Honda Controller 2	67	27	50	120	205	300	415
Wavelet Filter	30	16	20	30	55	80	105
DAC	354	108	300	480	775	1100	1690
modem	227	45	180	210	300	515	880
adaptive modem	200	48	200	510	600	830	1045
Large Controller	324	49	95	325	395	800	1250
LMS Audio Formatter	464	122	815	1435	2340	2880	4520
CORDIC	827	65	2050	3895	4670	5760	7580
Speech Compressor	1257	45	2895	5845	6475	7980	9220
2D Wang DCT	752	104	845	1560	2860	3985	5950
2D Arai DCT	1704	128	4865	8160	13520	18910	24235
2D Lee DCT	2048	184	3675	6520	9975	13635	18170

Table VII. The Overhead for $P_C = 10^{-50}$ and $P_T = 10^{-50}$ for the Designs in Table VI, Assuming the Edge Probability to be 0.3

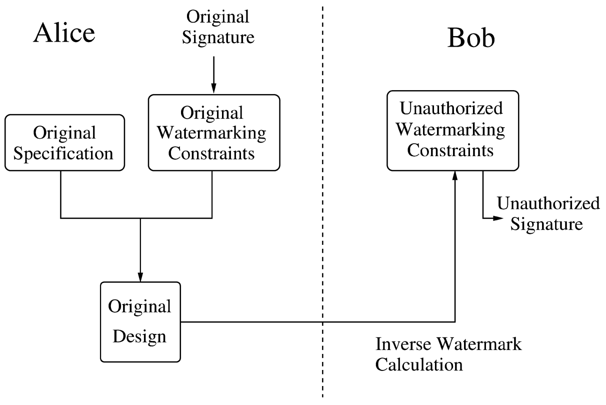
Graph	# of Bits in Signature	# of Bit Errors to be Corrected	# of Parity Bits for Error-Correcting Codes	# of Extra Colors Used
GE Controller 1	255	20	144	6
GE Controller 2	255	23	156	2
Honda Controller 1	255	27	172	3
Honda Controller 2	255	30	184	3
Wavelet Filter	127	21	98	5
DAC	511	26	234	2
modem	511	28	252	3
adaptive modem	511	25	225	2
Large Controller	1023	24	235	4
LMS Audio Formatter	1023	16	160	1
CORDIC	2047	18	198	1
Speech Compressor	2047	20	220	0
2D Wang DCT	2047	18	198	2
2D Arai DCT	4095	20	240	0
2D Lee DCT	8091	18	234	1

8. INVALIDATING FALSE CLAIMS OF OWNERSHIP

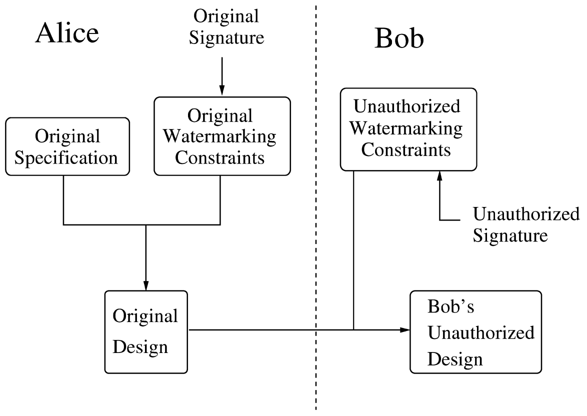
Consider the following scenario. Alice has a design D which is watermarked by the proposed watermarking scheme. Bob purchases the design D from Alice. There are three cases to consider. Each case is considered in the following subsections.

8.1 Finding Unintended Signatures

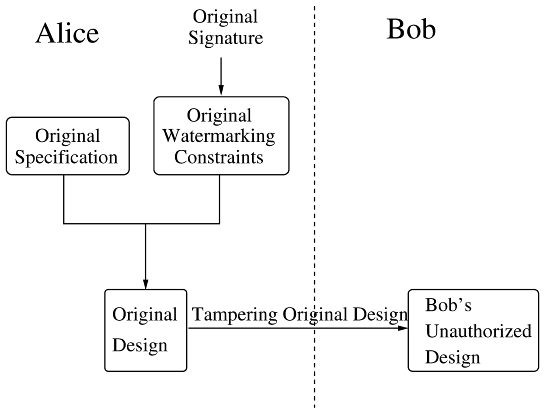
Bob tries to find his signature in the design D, as described in Figure 4(a). Bob claims that D is his design because D has both his and Alice's signatures. In this case, the one with the more meaningful, longer, and stronger signature


Alice
Original Signature
Original Specification
Original Watermarking Constraints
Original Design
Bob
Unauthorized Watermarking Constraints
Unauthorized Signature
Inverse Watermark Calculation

(a) Finding new signature


Alice
Original Signature
Original Specification
Original Watermarking Constraints
Original Design
Bob
Unauthorized Watermarking Constraints
Unauthorized Signature
Bob's Unauthorized Design

(b) Embedding new signature


Alice
Original Signature
Original Specification
Original Watermarking Constraints
Original Design
Tampering Original Design
Bob
Bob's Unauthorized Design

(c) Tampering original signature

Fig. 4. Schemes for breaking a watermark.

will be the winner. It is believed to be a very difficult task to find any meaningful signature from the design. It is much easier to embed the signature in a preprocessing step. Therefore, Alice can protect her authorship by embedding a strong watermark.

8.2 Embedding Unauthorized Signatures

Bob embeds his signature in the design D, as described in Figure 4(b), and claims that D is his design. In this case, Alice can easily prove that the design belongs to her. Alice's design has only her signature, while Bob's design has both his and Alice's signatures. Therefore, Alice can prove that it is her design. Note that this type of attack is similar to the one described in Craver et al. 1996.

8.3 Tampering with Original Signatures

In this case, Bob tries to tamper with Alice's signature by applying local changes, as described in Figure 4(c). There are at least three reasons that discourage this kind of attack. First, these local changes can result in worse design with more hardware overhead. Second, error-correcting and detection codes allow the design to withstand a significant amount of attack. Third, once Bob modifies the design, he has to perform all synthesis tasks below the affected part.

9. CONCLUSION

We introduced the first dynamic watermarking technique for protecting the value of intellectual property of CAD tools and reusable core components. The essence of the new approach is the addition of a set of design and timing constraints which encodes the author's signature. We established the first set of relevant metrics for the quantitative analysis of watermarking techniques which forms the basis for the quantitative analysis, evaluation, and comparison of watermarking techniques. We developed a generic approach for steganography and use it as a basis for the derivation of a spectrum of techniques for signature data hiding in designs. On a large set of design examples, experimental studies indicate the effectiveness of the new approach in the sense that highly resilient, difficult to detect and remove signatures are embedded in the designs with very low hardware overhead.

REFERENCES

Anderson, R. and Kuhn, M. 1996. Tamper resistance—A cautionary note. In *2nd USENIX Workshop on Electronic Commerce*, 1–11.

Bender, W., Gruhl, D., Morimoto, N., and Lu, A. 1996. Techniques for data hiding. *IBM Syst. J. 35*, 3/4, 313–336.

Benedens, O. and Busch, C. 2000. Towards blind detection of robust watermarks in polygonal models. In *Proceedings of the European Association for Computer Graphics* (*EUROGRAPHICS*) vol. 19, 199–208.

Berghel, H. and O'Gorman, L. 1996. Protecting ownership rights through digital watermarking. *IEEE Comput. 29*, 7, 101–103.

Boney, L., Tewfik, A. H., and Hamdy, K. N. 1996. Digital watermarks for audio signals. In *Proceedings of the International Conference on Multimedia Computing and Systems*. 473–480.

Brassil, J. T., Low, S., and Maxemchuk, N. F. 1999. Copyright protection for the electronic distribution of text documents. *Proceedings of the IEEE, 87*, 7, 1181–1196.

CHAPMAN, R. AND DURRANI, T. S. 2000. Ip protection of dsp algorithms for system on chip implementation. *IEEE Trans. Signal Process. 48*, 3, 854–861.

CHARBON, E. 1998. Hierarchical watermarking in ic design. In *Proceedings of the Custom Integrated Circuits Conference*. 295–235.

COX, I. J., KILIAN, J., LEIGHTON, T., AND SHAMOON, T. 1996. A secure, imperceptible yet perceptually salient, spread spectrum watermark for multimedia. In *Proceedings of IEEE Southcon*. 192–197.

CRAVER, S., MEMON, N., YEO, B. L., AND YEUNG, M. M. 1996. Can invisible watermarks resolve rightful ownerships? IBM Tech. rep., RC 20509, 1996.

CROCHIERE, R. E. AND OPPENHEIM, A. V. 1975. Analysis of linear digital networks. In *Proceedings of the IEEE 63*, 4, 581–595.

FERNANDEZ, D. 1994. Intellectual property protection in the EDA industry. In *Proceedings of the Design Automation Conference*. 161–163.

GIRCZYC, E. AND CARLSON, S. 1993. Increasing design quality and engineering productivity through design reuse. In *Proceedings of the Design Automation Conference*. 48–53.

HADA, S. 2000. Zero-knowledge and code obfuscation. In *Proceedings of the (ASIACRYPT) International Conference on the Theory and Application of Cryptology and Information Security*, T. Okamoto Ed., 443–457.

HARTUNG, F. AND KUTTER, M. 1999. Multimedia watermarking techniques. In *Proceedings of the IEEE 87*, 1079–1107.

IRBY, D. L., NEWBOULD, R. D., CAROTHERS, J. D., AND RODRIGUEZ, J. J. 2001. Placement watermarking of standard-cell designs. In *IEEE Southwest Symposium on Mixed-Signal Design*. 116–120.

JOHNSON, D. S., ARAGON, C. R., MCGEOCH, L. A., AND SCHEVON, C. 1991. Optimization by simulated annealing: An experimental evaluation, Part II. *Operat. Res. 39*, 3, 378–406.

JOHNSON, N. F., DURIC, Z., AND JAJODIA, S. 2001. *Information Hiding: Steganography and Watermarking—Attacks and Countermeasures*. Kluwer Academic, Boston, MA.

KAHNG, A. B., LACH, J., MANGIONE-SMITH, W. H., AND MANTIK, S. 2001. Constraint-based watermarking techniques for design ip protection. *IEEE Trans. Comput.-Aid. Design Integrat. Circuits Syst. 20*, 10, 1236–1252.

KATZENBEISSER, S. AND PETITCOLAS, F. A. P. EDS. 2000. *Information Hiding Techniques for Steganography and Digital Watermarking*. Artech House, Boston, MA.

KHANNA, S. AND ZANE, F. 2000. Watermarking maps: hiding information in structured data. In *ACM/SIAM Symposium on Discrete Algorithms* (*SODA*), 596–605.

KIM, H. J., MANGIONE-SMITH, W. H., AND POTKONJAK, M. 1998. Protecting ownership rights of a lossless image coder through hierarchical watermarking. In *Workshop on Signal Processing Systems*, 73–82.

KIROVSKI, D., HWANG, Y., POTKONJAK, M., AND CONG, J. 1998. Intellectual property protection by watermarking combinational logic synthesis solutions. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. 194–198.

KIROVSKI, D. AND MALVAR, H. S. 2001. Spread-spectrum audio watermarking: Requirements, applications, and limitations. In *IEEE International Workshop on Multimedia Signal Processing*.

KIROVSKI, D. AND POTKONJAK, M. 2003. Local watermarks: Methodology and application to behavioral synthesis. *IEEE Trans. VLSI CAD 22*, 9, 1277–1284.

KOCHER, P. C. 1996. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. Lecture Notes in Computer Science, *Advances in Cryptology, CRYPTO*, 104–113.

KOCHER, P. C. 1999. Differential power analysis: Leaking secrets. Lecture Notes in Computer Science, 388–397.

KOUSHANFAR, F., HONG, I., AND POTKONJAK, M. 2003. Behavioral synthesis techniques for intellectual property protection. Tech. Rep., UCLA Computer Science Department.

KOUSHANFAR, F., QU, G., AND POTKONJAK, M. 2001. Intellectual property metering. In *IHW Information Hiding Workshop*, 2137, 87–102.

KUMAGAI, J. 2000. Chip detectives [reverse engineering]. *IEEE Spectrum 37*, 11, 43–48.

KUTTER, M. AND WINKLER, S. A. 2002. A vision-based masking model for spread-spectrum image watermarking. *IEEE Trans. Image Process. 11*, 1, 16–25.

LACH, J., MANGIONE-SMITH, W. H., AND POTKONJAK, M. 1998. Fingerprinting digital circuits on programmable hardware. *Information Hiding Workshop*. 16–31.

LACH, J., MANGIONE-SMITH, W. H., AND POTKONJAK, M. 2001. Fingerprinting techniques for field-programmable gate array intellectual property protection. *IEEE Trans. Comput.-Aid. Design Integrat. Circuits Syst. 20*, 10, 1253–1261.

LEE, E. A. AND MESSERSCHMITT, D. G. 1987. Synchronous dataflow. In *Proceedings of the IEEE 75*, 9, 1235–1245.

LIN, S. AND COSTELLO, D. J. 1983. *Error Control Coding*. Prentice Hall.

LOFSTROM, K., DAASCH, W. R., AND TAYLOR, D. 2000. IC identification circuit using device mismatch. In *Proceedings of the IEEE International Solid-State Circuits Conference*. 372–373.

MEGUERDICHIAN, S. AND POTKONJAK, M. 2000. Watermarking while preserving the critical path. In *Design Automation Conference*. 108–111.

DE MICHELI, G. 1994. *Synthesis and Optimization of Digital Circuits*. McGraw-Hill, New York, NY.

NEWBOULD, R. D., CAROTHERS, J. D., AND RODRIGUEZ, J. J. 2002. Watermarking ics for ip protection. *Electron. Letters 38*, 6, 272–274.

NEWBOULD, R. D., IRBY, D. L., CAROTHERS, J. D., AND RODRIGUEZ, J. J. 2001. Mixed signal design watermarking for ip protection. In *IEEE Southwest Symposium on Mixed-Signal Design*. 110–115.

OHBUCHI, R., MASUDA, H., AND AONO, M. 2000. A shape-preserving data embedding algorithm for nurbs curves and surfaces. *Trans. Inform. Process. Soc. Japan 41*, 3, 559–569.

OLIVEIRA, A. L. 2001. Techniques for the creation of digital watermarks in sequential circuit designs. *IEEE Trans. Comput.-Aid. Design Integrat Circuits Syst. 20*, 9, 1101–1117.

PODILCHUK, C. AND ZENG, W. 1997. Perceptual watermarking of still images. In *IEEE Workshop on Multimedia Signal Processing*. 363–368.

RABAEY, J., CHU, C., HOANG, P., AND POTKONJAK, M. 1991. Fast prototyping of data path intensive architectures. *IEEE Design and Test of Comput. 8*, 2, 40–51.

RASHID, A., ASHER, J., MANGIONE-SMITH, W. H., AND POTKONJAK, M. 1999. Hierarchical watermarking for protection of dsp filter cores. In *Custom Integrated Circuits Conference*. 39–42.

SCHNEIER, B. 1996. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley and Sons, New York, NY.

SWANSON, M. D., ZHU, B., CHAU, B., AND TEWFIK, A. H. 1997. Object-based transparent video watermarking. In *IEEE Workshop on Multimedia Signal Processing*. 369–374.

TORUNOGLU, I. AND CHARBON, E. 1999. Watermarking-based copyright protection of sequential functions. In *Custom Integrated Circuits Conference*.

VAN SCHYNDEL, R. G., TIRKEL, A. Z., AND OSBORNE, C. F. 1994. A digital watermark. In *Proceedings of the International Conference on Image Processing*. 86–90.

WAGNER, D., FOSTER, J. S., BREWER, E. A., AND AIKEN, A. 2000. A first step towards automated detection of buffer overrun vulnerabilities. In *Network and Distributed System Security Symposium*.

WONG, J. L., KIROVSKI, D., AND POTKONJAK, M. 2001. Computational forensic techniques for intellectual property protection. In *Information Hiding Workshop*, 2137, 71–86.

YEO, B. AND YEUNG, M. M. 1999. Watermarking 3d objects for verification. *IEEE Comput. Graph. Applica. 19*, 1, 36–45.

YEUNG, M. M., MINTZER, F. C., BRAUDAWAY, G. W., AND RAO, A. R. 1997. Digital watermarking for high-quality imaging. In *IEEE Workshop on Multimedia Signal Processing*. 357–362.

Received January 2003; revised October 2003, May 2004; accepted June 2004