IP Watermarking Techniques: Survey and Comparison

Amr T. Abdel-Hamid*, Sofiène Tahar* and El Mostapha Aboulhamid§
*Electrical and Computer Engineering Department, Concordia University, Montréal, Canada
Email: {at_abdel, tahar}@ece.concordia.ca
§Dep. d'informatique et de recherche operationnelle, Université de Montréal, Montréal, Canada
Email: aboulham@iro.umontreal.ca

***Abstract*— Intellectual property (IP) block reuse is essential for facilitating the design process of System-on-a-Chip. Sharing IP blocks in such a competitive market poses significant high security risks. IPs can be read, copied or even partitioned to cover the authorship proof. Creators and owners of IP designs want assurance that their content will not be illegally redistributed by consumers. Consumers, on the other hand, want assurance that the content they buy is legitimate. Digital watermarking, used with most of the shared digital media, has emerged as a candidate solution for helping copyright protection of IP blocks. In this paper, we outline IP watermarking and survey the current state-of-the-art of different schemes and algorithms. We also highlight the main technical problems that should be solved in order to let IP watermarking be used widely in industry.**

I. Introduction

Fast advancing IC (integrated circuit) processing technologies have enabled the integration of full systems an a single chip forming the new paradigm of the "System-on-a-Chip" (SOC) technology. Incremental changes to current design methodologies are inadequate for enabling full potential SOC implementation. As proposed in [2], the required shift needed for SOC design rests on two main industrial trends: The wide availability of reusable virtual components, and, the development of application-oriented IC integration platform to reduce development time and efforts. Reusable virtual components or *intellectual property* (IP) blocks are most effective when coming to reducing cost and development time of SOC designs.

Sharing IP designs poses significant high security risks. Most of these IPs need time and effort to be designed and verified, yet they can be easily copied, or modified to cover the authorship proof. Creators and owners of IP designs want assurance that their content will not be illegally redistributed by consumers, and consumers want assurance that the content they buy is legitimate.

Throughout history, *watermarking* was widely used for copyright protection as well as data hiding. Recently, digital watermarking has emerged as a candidate solution for the copyright protection problem of digital media (such as video, pictures, or music). IP watermarking is being also introduced as a candidate to protect this sensitive copyright information.

A. System-on-a-Chip (SOC): Design Path

The SOC revolution did not only add new functionalities and larger systems, but also changed the way such systems used to be designed. Integration between hardware and software from the early design stages is becoming essential to insure better performance and functionality. Co-design, and co-verification approaches are needed to enable the design and implementation of such systems. According to [2], a System-on-a-Chip is defined as "*a complex IC that integrates the major functional elements of a complete end-product into a single chip or a chipset*".

An SOC design process starts at the system level (Figure 1), where different high level aspects, like specifications and requirements are delivered [1]. The system level model is designed by introducing the requirements in both the architectural and algorithmic designs [2]. In the algorithmic (functional) design, the product requirements are established and a verified specification of the system's function is produced. This level results in the main functional specification of the system needed to be implemented. Afterwards at the architectural level, the system specification is decomposed and mapped into architectural blocks according to the algorithmic design. In this level, the architecture or a family of architectures on which the system will be realized is defined. These architectures include components, such as microprocessors, memory components, operating systems. The behavioral models, of both software and hardware, are generated by assigning every function to a specific hardware or software resource. This process results in the behavioral specifications of the system.

The partitioned modules are developed separately in the lower levels, yet the specification is used for mutual testing and simulation during different implementation levels. The software part is developed using different programming languages that are compatible with the hardware afterwards. In the hardware part, the design takes the same hierarchical approach usually used in the hardware design path (Figure 1).

To facilitate and enhance such long design process, IP blocks are used at different hardware/software design levels. IP blocks, either reusable or those needed to be designed, are defined in the architectural level. These pre-designed, pre-tested IP blocks allows system engineers to make necessary modifications and meet users requirements in a timely fashion.

B. IP Blocks and Distribution Threats

IP blocks are delivered in three main flavors depending on price, applications, and contracts between companies. The

Proceedings of The 3rd IEEE International Workshop on System-on-Chip for Real-Time Applications ISBN 0-7695-1929-6/03 $17.00 © 2003 IEEE

IEEE COMPUTER SOCIETY

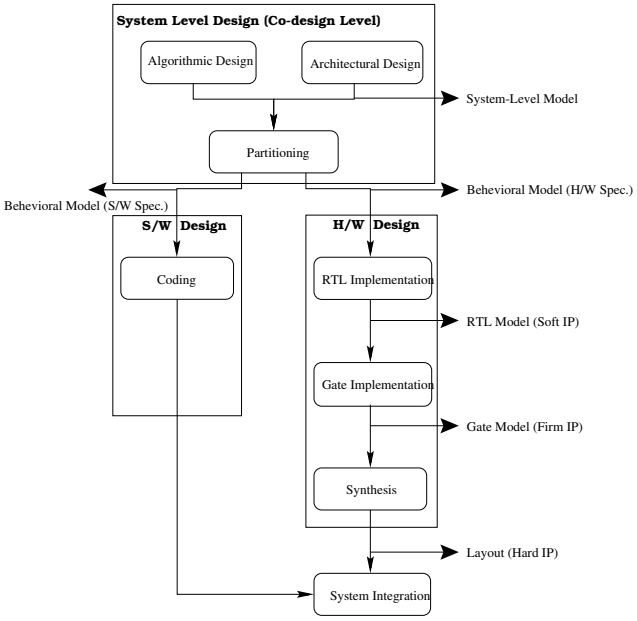

Fig. 1. SOC Design Flow and Different Forms of IP Blocks

Virtual Socket Interface (VSI) architecture document [19] describes such levels as (Figure 1):

Soft IP: are delivered in the form of synthesizable hardware design language (HDL), i.e., high level designs. They have the advantage of being more flexible and the disadvantage of not being as predictable in terms of performance (e.g., timing, area, power). Soft IPs typically have increased Intellectual Property risks because RTL (register transfer level) source code is required by the integrator.

Firm IP: have been optimized in structure and in topology for performance and area through floor planning/placement, possibly using a generic technology library. Firm IPs offer a compromise between Soft and Hard. More flexible and portable than Hard, yet more predictive of performance and area than Soft. Firm IPs include a combination of synthesizable RTL, reference technology library, detailed floorplan, and a full or partial netlist. Firm IPs do not include routing. Risks are equivalent to those of Soft IPs if RTL is included and are less if it is not included.

Hard IP: have been optimized for power, size, or performance and mapped to a specific technology. Examples include netlists that are fully placed, and routed, or optimized custom physical layout. They have the advantage of being much more predictable, but consequently are less flexible and portable due to process dependencies. Hard IPs require, at a minimum, a high level behavioral model, a test list, full physical and timing models along with the final layout. The ability to protect Hard IPs is much better because of copyright facilities and there is no requirement for an RTL code to be included.

The VSI Alliance IP protection development working group [8] identifies three main approaches to secure IPs. First, a *deterrent* approach where the owner uses legal means trying to stop attempts for illegal distribution, i.e., using patents, copyrights and trade secrets. This method does not provide any physical protection to the IP. Second, a *protection* approach where the owner tries to prevent the unauthorized usage of the IP physically by license agreements and encryption. Third, a *detection* approach where the owner detects and traces both legal and illegal usages of the designs as in watermarking and fingerprinting. This tracking should be clear enough to be considered as evidence in front of a court if needed. The VSI alliance proposed the usage of the three approaches for proper protection of IP designs. The detection approach directly interacts with the system design, and is considered an overhead on the design cycle. IP *watermarking* and IP *fingerprinting* are the main approaches used for detection.

In this paper, we outline IP watermarking and fingerprinting techniques for copyright protection, surveying the current state-of-the-art of IP digital watermarking research. Also, we highlight the main technical problems that must be solved before digital watermarking can be widely used in industry.

The rest of the paper is organized as follows: Section II describes briefly the preliminaries of digital watermarking and the main attack classes against watermarking. Section III overviews the main approaches used for IP watermarking. It describes the main advantages and disadvantages of each technique as well as a comparison between them. Finally, Section IV extracts the main guidelines and conclusions for future IP watermarking development directions.

II. Digital watermarking

In [9], Kahn mentioned many examples throughout history about hiding data on another media as a cover. The author tells about a prisoner of war who hides messages in letters to home using the dots and dashes on i, j, t and f to spell out a hidden text in Morse code. As another example, the clear signature of most of the famous artists on their paintings was one of the first copyright watermarks ever known.

In their survey about data hiding techniques, Petitcolas *et al.* [15] defined the term *steganography* as "*having a covert communication between two parties whose existence is unknown to a possible attacker*". Steganography is divided into three main application classes. First, *information hiding*, which utilize the secrecy and undetectability of steganography to transfer secret data without detection, used mainly for espionage applications. Second, *content verification* applications (*authentication*), where a fragile watermark is introduced to secure the contents integrity. Finally, *intellectual property* protection applications, where the watermark is mainly used to convey the information about content ownership and intellectual property rights.

Copyright marking (widely known as watermarking), as opposed to steganography, has the additional requirement of robustness against possible attacks. Robust watermarking have the property of being infeasible to remove them or make them useless without destroying the object at the same time. This means that usually it has to be embedded in the most perceptually significant components of the object [15].

Proceedings of The 3rd IEEE International Workshop on System-on-Chip
for Real-Time Applications ISBN 0-7695-1929-6/03 $17.00 © 2003 IEEE

IEEE COMPUTER SOCIETY

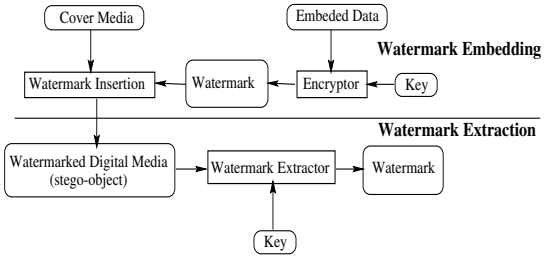

Fig. 2. Digital Watermarking

Figure 2 describes a generic model of watermarking [6]. The process is divided into two parts: *watermarking embedding*, and *watermark extraction* (also known as *tagging* and *tracking*, respectively). In the embedding phase, the embedded data, which is the message that one wishes to send secretly, is usually hidden in another media referred to as a cover-text, or cover-image (in our case cover-code or cover-media). This produces the stego-text or other *stego-object*. A key (*stego-key*) is used to control the hiding process, thus restricting detection and/or recovery of the embedded data to parties who know it (or who know some derived key value). This stego-key can be either a public key or a private key depending on the scheme of the watermarking. In the extraction phase, the stego-object is used with the key to extract the watermark and identifies it. The robustness and strength of a watermark are measured usually using benchmarks, e.g., Stirmark [14] for images.

III. IP Watermarking: State-of-the-Art

IP watermarking is comparatively a new field. In this survey, our aim is to discuss these approaches, exposing the main advantages and disadvantages of each, as well as, the different attack schemes that the intruder might. We will start by categorizing the main attacks against IP watermarking in the next subsection as well as different intruder schemes. There are three main watermarking schemes discussed in the open literature: *Constraint-Based* watermarking, *Finite State Machines* watermarking, and *Digital Signal Processing* watermarking. Each associated with one or more algorithm or technique, which will be discussed in the subsections to follow.

A. Attacks Against IP Watermarking

Some of the essential aspects (may be even the most essential one) that have to be studied with any security system, are attacks and risks against such techniques. Digital watermarking attacks are categorized in four main classes [6]: *unauthorized removal*, *unauthorized embedding*, *unauthorized detection*, and *system attacks*. The same categorization applies for IP watermarking schemes, yet unauthorized detection is not considered a high risk for IPs.

Removal attacks aim at the removal of the watermark information [18]. This is tried without breaking the security of the watermark, i.e., without searching for the key used in the embedding. Removal attacks are divided into either elimination attacks or masking attacks. The intruder tries to eliminate the watermark completely in the elimination attacks. As an example, the intruder tries to estimate the watermark and subtract it from the watermarked design. On the other hand, masking attacks do not aim at removing the watermark itself, but rather aim at distorting the watermark detector such that it will not be able to sense the availability of the watermark. This attack is considered one of the main measures for robustness of a watermark. Collision attacks, de-noising, compression and demodulation are used in such attacks in case of multimedia objects (or data). In IP watermarking, finite state machine reduction for instance can be used in some cases to delete the watermark.

Embedding attacks (forging) aim at embedding another watermark in the design, this can be done either by ghost searching, where the intruder tries to find a ghost watermark and consider it as his watermark, or by re-embedding the watermark if he/she has the tools necessary to do this.

System attacks aim on attacking the concept of watermarking its self, as an example, attacking the cryptographic base of the watermarking, or removing the chip that check the watermark physically in case of video for instance. This kind of attacks cannot to be avoided by the watermarking schemes, yet the VISA IP protection group solves this by protecting the design through different transactions, i.e., using protection techniques to overcome such problem.

B. Constraint-Based IP Watermarking

Kahng *el al.* [10] proposed the *constraint-based* IP watermarking as one of the leading approaches for IP watermarking. This approach is a generic algorithm that can be used at different levels of the design flow.

it is based on the usage of available tools used mainly to solve NP-hard problems. The algorithm adds extra constraints to such solutions that would make it yield the new watermarked design as discussed in details below. The approach is based on a generic optimizer and the constraint-satisfaction (SAT) problems. The watermarking tool proposed is mainly composed of the following parts (Figure 3):

1) An optimization problem, which is an NP-hard problem that needs constraints and heuristics to be solved.

2) An off-the-shelf optimization software/algorithm to solve such a problem.

3) A set of constraints that should be applied to the design.

4) A well-formed grammar to add extra-constraints to the previous ones for building the required watermarked design. This is the main watermarking tool, it is composed of a one-way encryption functions that convert the watermark of the code to a set of well-formed constraints.

The watermark is presented to the constraint generator, where it is first encrypted, then transferred through a hash function (to shorten its length). Finally, it is converted to a set of extra constraints, through the well-formed grammar, forming a new set of constraints which is added to the available ones. Both the design and the set of constraints are feed to the black-box optimizer resulting in a watermarked solution.

Proceedings of The 3rd IEEE International Workshop on System-on-Chip
for Real-Time Applications ISBN 0-7695-1929-6/03 $17.00 © 2003 IEEE

IEEE COMPUTER SOCIETY

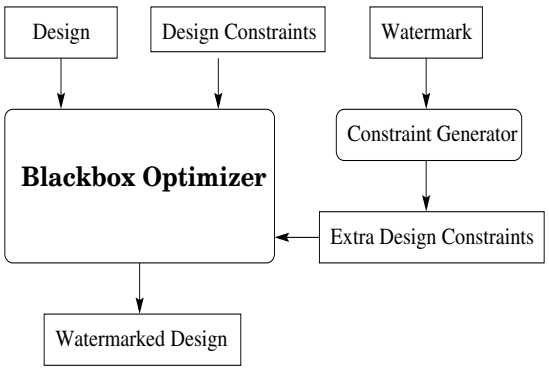

Fig. 3. Constraint-Based IP Watermarking

The watermark is then a set of extra constraints that will limit the set of possible solutions to a smaller set. The watermark becomes stronger as the "*watermark subset*" is smaller.

Two factors are used to measure the strength of this approach: (1) The cost of the optimization problem, where the harder the solution is, the stronger and more meaningful the watermark, as resolving it will cost nearly as much as the re-design of the whole project; (2) The probability (P_c) that a non-watermarked design would carry this watermark by coincidence. Although this is a main term of the strength, it is not calculated exactly, however an upper bound assuming all constraints to be independent was calculated.

Due to the generic nature of the approach, it was tested and applied to different levels of the IP design. It was extended for most of the design levels NP-hard problems. At the system level, it was used for watermarking of memory graph coloring solutions by Hong *el al.* [7], as well as using it in solving linear programming problems by Megrian *et al.* [11]. At lower design levels, the physical level for instance, the approach was used even more effectively as it was linked to the NP-hard problems available at such levels such as routing [12].

The main advantage of the approach is its real low overhead, as the NP-hard problem will be solved anyway as well as the constraint generator does not cause much overhead. Nevertheless, the approach has some major drawbacks. First, the watermark cannot be detected except at the same level of abstraction, i.e., tracking the watermark is not that easy if the design is resold at other abstraction levels. The algorithm is missing a good tracking way at lower design levels. Another main drawback is the fact that imposing extra-constraints on a design procedure might not be as successful as thought by the authors. For instance, some constraints might contradict, or at least pull the solution away from the optimized solution expected. In this case, we will need to redo the whole process of changing our signature to get better results. This might introduce a high overhead in the design process and might not yield to satisfactory results in some cases. Finally, exposing the well-formed grammar in any legal dispute, would weaken other watermarked designs especially against forging attacks. This means that the algorithm in a way is dependent on the secrecy of the defined grammar.

The generic approach though proved to be immune for most of the available attacks. The idea of injecting the watermark in a non-linear problem by nature gives it this real high strength. Removal attacks would need to resolve the NP-hard problem, which mostly would cost as much as rebuilding the whole design again, especially if the intruder needs to re-engineer the design. Forging, by either trying to add the intruder signature or finding a ghost signature highly depends on the probability P_c which was proved to reach 2^{-56} in some cases.

C. Watermarking Finite State Machines

At the behavioral level, Oliveira [13], and Torunoglu *et al.* [17] introduced two different techniques used in the watermarking of sequential parts of the design. Both algorithms are based on adding new input/output sequences to the finite state machines (FSM) representation of the design. The main advantage of both approaches is the ability to detect the presence of the watermark at all lower design levels.

1) FSM Watermarking Based on Unused Transitions: Torunoglu and Charbon [17] introduced the first approach on FSM watermarking. The algorithm is mainly based on extracting the unused transitions in a state transition graph (STG) of the behavioral model. These unused transitions are inserted in the STG associated with a new defined input/output sequence, which will act as the watermark.

The approach in [17] starts with building the FSM representation of the function, then visiting every state and finding the unused state transitions (input/output pairs). In case the FSM is completely specified (CSFSM), new input/output pairs are added to expand the FSM. The minimum number of transitions needed is then calculated, and compared to the maximum number of free transitions to satisfy the probability (P_c) that a non-watermarked design would carry this watermark by coincidence. If this probability cannot be satisfied, input/output pairs should be added to satisfy the watermark requirements.

The input/output sequence is calculated, such that the input sequence is random to the set of unused transition inputs. On the other hand, the output, which is the hidden information, is encrypted using a key. Extra transitions are added such that the output of the given input sequence generates the encrypted hidden data, i.e., one should have both the key and the input sequence to be able to read the watermark.

Figure 4 [17] shows an example of the watermarking process, where Figure 4 (a) shows the original design, Figure 4 (b) describes the watermarked design, and Figure 4 (c) shows another watermarked solution after augmenting the inputs to add more transitions. The strength of such process is related mainly to the same P_c factor described in Section III-2. The authors in [17] have proven an upper bound for this probability depending on the number of free transitions used and the number of free transitions needed.

The approach is very promising, as it works at a very high level which provides extra strength. Also, it can be detected at mostly all lower levels sometimes even after design manufacturing. The approach still has some drawbacks:

IEEE COMPUTER SOCIETY

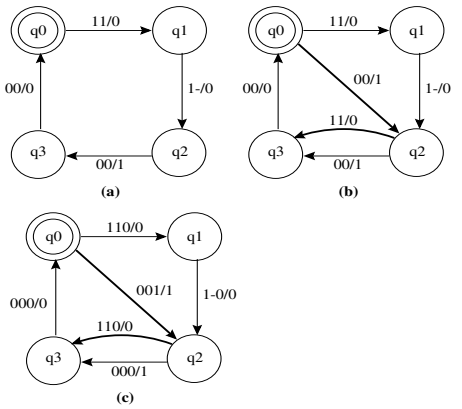

Fig. 4. FSM Watermarking Based on Unused Transitions

(1) Finding the input sequence that satisfies P_c and is not considered with a high overhead on the STG is an NP-hard problem. The authors of [17] proposed exhaustive search, or Monte-Carlo search [4] to get over this, yet solving this would propose a high overhead in the design phase; (2) The algorithm is mechanically removable in case of the knowledge of the input sequence. So, public-key watermarking cannot be used with this approach because an intruder who knows the input sequence of the watermark, can build a machine (program) that will delete all such transitions from the FSM.

The algorithm is immune for FSM reduction techniques, as the variables used are usually in other transitions, which makes it real hard to remove. Yet, if the intruder has the STG but does not have the input/output mapping (the case of most IP blocks), he/she might try to tamper the watermarking, by changing the output of the next state related to the transition. This may result in a change in functionality. Besides, the intruder might try to read his/her own watermark using the same concept. The authors in [17] tried to solve this by building a machine based on the Genome search [17] to find the tampered watermark.

2) FSM Watermarking by Property Implanting: Oliveira [13] proposed the implicit manipulation of the STG of the finite state machine to implant the watermark as a property in the new one. To watermark a design, the user should define an arbitrary long string that clearly describes his/her ownership rights. This data is considered the watermark information. After encrypting this message using a public key, the user should then use a one-way hash function, such as MD5 [16], to obtain a compact signature of this arbitrarily long sentence. The arbitrary sequence is then broken to input sequence combinations. For example, if the design has 16 inputs, and the sequence is 128 bits, it defines a unique sequence of 8 input combinations.

The user then changes the STG in such a way that the sequence of states reached by this sequence of inputs exhibits a specific property, which is rare in non-modified STGs. This property is purely topological and does not depend on the specific encoding. If, later on, the watermark need to be uncovered, the designer shows this input sequence and the property he/she defined.

In order to define the input sequence to change the STG properties, Oliveira [13] adds extra states and transitions in a systematic way to satisfy this property. The algorithm has a low overhead on the design flow, because it does not need to go through the FSM to find the unused transitions (an NP-hard problem as discussed above). In [13], it is even proposed to use a very strong way to build and implant his watermark without the need of building the FSM of the design.

The approach depends on adding a counter that checks for the input sequence expected and it reaches a certain value to indicate that the design has traversed the watermark proposed. This counter is an internal signal that can only be seen during simulation of the design, i.e., it cannot be detected from outside. The author of [13] as well, did not calculate the probability P_c in any of the designs. Instead, he relied on another probability P_{beta}, which was defined as false positive detection. The extra states added can be removed sometimes using a state reduction approach, although the author proposed to solve this problem by slightly changing the functionality of the STG. This can never be done mechanically as it is pretty complicated and might affect the design functionality.

D. Digital Signal Processing Watermarking

At the algorithmic level, Chapman *et al.* [3] proposed a Digital Signal Processing (DSP) watermarking scheme. The algorithm is based on the ability of designers to make minor changes in the decibel (db) requirements of filters.

In this approach, the designer of a high level digital filter should encode one character (7 bits) as his/her hidden watermark data. Then the high level filter design is divided into 7 partitions where each partition is used as a modulation signal of one of the bits. This means dividing the filter to 7 parts and use each part as a carrier signal with little db change if the bit is one or no db change if the bit it zero.

This approach can be used at the DSP design level, which is a real high level design, yet the authors of [3] did not discuss the strength of their approach or the probability P_c that this bits might coincide with another un-watermarked design. The approach as well depends on a very low data rate, just one character (7 bits), which makes it really unpractical to be used in an industrial environment. The approach also is missing a clear way to track and extract the watermark at lower levels.

IV. IP Watermarking: Evaluation and Comparison

Cox *et al.* [6] described the main properties that would judge the suitability of a watermarking scheme for a certain application. From their property list, we have chosen those mostly applicable to IP and would affect mostly the decision when choosing an IP watermarking technique.

Table I shows the five main properties we chose for watermarking evaluation. We started by the *embedding cost*, which expresses both the combinational cost of the embedder and the time needed for embedding such watermark. The *overhead* a watermark adds to the design compared to the actual design. The ability and cost of *tracking* the watermarked

IEEE COMPUTER SOCIETY

TABLE I
COMPARISON OF DIFFERENT IP WATERMARKING APPROACHES

Property	Constraint based	Unused Transitions	Property Implanting	DSP
Embedding Cost	Low	High	Medium	Low
Overhead	Low	Low	Medium	Low
Tracking Cost	High	Low	Low	High
Probability of Coincidence	Low	Low	Low	High
Security	High	Medium	Medium	High

design afterwards. The *probability of coincidence*, calculated by P_c, giving the probability a non-watermarked design might coincide by accident with a watermarked one. Finally, the *security*, which is the ability of the watermark to resist different hostile attacks. The robustness and the strength of a watermark are usually measured using benchmarks. For instance, Stirmark [14] is one of the main benchmarks for image watermarking techniques. For IP watermarking, there is no benchmark available so people use the probability of occurrence, P_c, and other probabilities to measure the strength of their watermarks.

It became clear from Table I, that the constraint-based watermarking technique has the advantage of low cost for both design overhead and embedding. Yet, it has a real drawback of the inability of tracking this approach through simple tracking tools. Also, the solution derived using this approach might shift from the optimum solution that can be generated without the extra added constraints.

Both of the FSM based approaches (adding either properties or making use of unused transitions) have problems on their own. FSM watermarking utilizing unused transitions search for all unused transitions can cause state space explosion easily and this is considered a real high embedding cost. While adding properties for the FSM seems more attractive in terms of embedding cost, yet its high design overhead, and vulnerability to reduction techniques is still considered a real problem as well. Both approaches have the very important advantage of being tractable all over the IP block life cycle (sometimes even after manufacturing).

Finally, the DSP watermarking approach has a lower design overhead and embedding cost as the addition of the watermark is nearly totaly transparent to the design process. Nevertheless, the amount of data inserted is very low (1 byte), this would cause a low P_c. Besides, it is very hard to be tracked in the output noise of the manufactured design.

Future IP watermarking schemes should be robust enough to secure the design, yet they should not imply a high overhead on neither the design process nor the final watermarked product. Adding a hierarchy of watermarks through the design cycle can give a more robust watermark against attacks [5]. Starting form high levels of the design (i.e., system level) and integrating the watermark through many design levels insures robustness, which decreases the risks of destroying the watermark. These watermarks should be easily detectable at lower design levels to insure proper tracking. Efficient watermark schemes should also use a Public-key encryption algorithm in the watermarking process, thus allowing third party entities (like brokers, testers) to get into the distribution cycle without security hazards. Finally, IP watermarking developers are missing a strong benchmark like those available, e.g., for photos. Such benchmark would be a balanced measure for the strength of different approaches. Benchmarking an IP watermarking scheme is harder than for instance of photos as the watermark might be spread in many design levels, given the different nature through the design span of SOC.

REFERENCES

[1] Cadence Corp., "Cadence System Level Design and Verification", Cadence Design Systems, White paper, 2002.

[2] H. Chang, L. Cooke, M. Hunt, G. Martin, A. McNelly, and L. Todd, "Surviving the SOC Revolution: A Guide to Platform-Based Design", Kluwer Academic Publishers, 1999.

[3] R. Chapman and T. S. Durrani, "IP Protection of DSP Algorithms for System on Chip Implementation", IEEE Transactions on Signal Processing, Vol. 48, No. 3, March 2000, pp. 854-861.

[4] E.D. Cashwell and C.J.Everett, "A Practical manual on the Monte Carlo Method for Random Walk Problems", Pergamon Press, 1959.

[5] E. Charbon, "Hierarchical Watermarking in IC Design", IEEE 1998 Custom Integrated Circuits Conference, Santa Clara, California, USA, May 1998, pp. 295 - 298.

[6] I. J. Cox, M. L. Miller, and J. A. Bloom, "Digital Watermarking", Morgan Kaufmann Publishers, ISBN 1-55860-714-5, 2002.

[7] I. Hong and M. Potkonjak, "Techniques for intellectual property protection of DSP designs", IEEE International Conference. Acoustics, Speech, and Signal Processing, Munich, Germany, vol. 5, 1998, pp. 3133-3136.

[8] Intellectual Property Protection Development Working Group, "Intellectual Property Protection: Schemes, Alternatives and Discussion", VSI Alliance, White Paper, Version 1.1, August 2001.

[9] D. Kahn, "The Codebreakers: The Story of Secret Writing", Scribner, 1996.

[10] A. B. Kahng, *et al.*, "Copy Detection for Intellectual Property Protection of VLSI Design" Proc. IEEE/ACM International Conference on Computer-Aided Design, San Jose, California, USA, November 1999, pp. 600-604.

[11] S. Megerian, M. Drinic, and M. Potkonjak. "Watermarking Integer Linear Programming Solutions", Proc. 39th IEEE/ACM Design Automation Conference, New Orleans, LA, USA, June 2002, pp. 8-13.

[12] N. Narayan, *et al.*, "IP Protection for VLSI Designs Via Watermarking of Routes," in Proc. 14th Annual IEEE International ASIC/SOC Conference, Washington, DC, USA, September 2001, pp. 406-410.

[13] A. L. Oliveira, "Techniques for the Creation of Digital Watermarks in Sequential Circuit Designs", IEEE Transactions om Computer-Aided Design of Integrated Circuits and Systems, Vol. 20, No. 9, September 2001, pp. 1101-1117.

[14] F. A. P. Petitcolas, R.J. Anderson, and M. G. Kuhn, "Attacks on copyright marking systems", Proceeding of Information Hiding, Second International Workshop, LNCS 1525, Springer-Verlag, April 1998, pp. 219-239.

[15] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Information Hiding- A Survey", Proceeding of the IEEE, special issue on the protection of multimedia content, Vol. 87, No. 7, July 1999, pp. 1062-1078.

[16] R. Rivest, "RFC 1321: The MD5 Message-Digest Algorithm", Network Working Group, 1992.

[17] I. Torunoglu, and E. Charbon, "Watermarking-Based Copyright Protection of Sequential Functions", IEEE Journal of Solid-State Circuits, Vol. 35, No. 3, February 2000, pp.434-440.

[18] F. Vahid, and T. Givargis, "Embedded System Design: A unified Hardware/Software Introduction", John Wiley & Sons, 2002.

[19] VSI Alliance, "VSI Alliance Architecture Document: Version 1.0", VSI Alliance, 1997.

Proceedings of The 3rd IEEE International Workshop on System-on-Chip for Real-Time Applications ISBN 0-7695-1929-6/03 $17.00 © 2003 IEEE

IEEE COMPUTER SOCIETY