VLSI Design IP Protection: Solutions, New Challenges, and Opportunities

Lin Yuan and Gang Qu

Electrical and Computer Engineering Department and Institute for Advanced Copmuter Studies
University of Maryland, College Park, MD U.S.A.
{yuanl, gangqu}@glue.umd.edu

Lahouari Ghouti and Ahmed Bouridane

School of Computer Science and the Institute of Electronics, Communications and Information Technology
Queen's University of Belfast, Belfast, UK
{L.Ghouti, a.bouridane}@qub.ac.uk

Abstract

It has been a decade since the need of VLSI design intellectual property (IP) protection was identified [1,2]. The goals of IP protection are 1) to enable IP providers to protect their IPs against unauthorized use, 2) to protect all types of design data used to produce and deliver IPs, 3) to detect the use of IPs, and 4) to trace the use of IPs [3]. There are significant advances from both industry and academic towards these goals. However, do we have solutions to achieve all these goals? What are the current state-of-the-art IP protection techniques? Do they meet the protection requirement designers sought for? What are the (new) challenges and is there any feasible answer to them in the foreseeable future?

This paper addresses these questions and provides possible solutions mainly from academia point of view. Several successful industry practice and ongoing efforts are also discussed briefly.

1. Introduction

The ever-increasing logic density has resulted in more transistors on silicon than designer's ability to design them meaningfully. This creates the *design productivity gap* between what can be built and what can be designed. Despite the design technology innovations in the last decade, this gap is becoming wider and wider. To close this gap, we need a significant shift in design methodology. At the heart of this shift is the principle of design reuse, which is the most significant design technology innovation in the past decade as one can see from Figure 1.

In this new design method, large previously designed blocks, such as bus controllers, CPUs, and memory subsystems, will be integrated into an ASIC architecture to deliver routine functions in a predictable manner. Designers now can focus on what they do best to design new blocks, representing their true design innovation, based on the system requirements. This not only makes it possible to have the new products on market in a timely and cost effective fashion, the newly developed blocks will also be tested, documented, and deposited as design intellectual properties (IPs) in the internal IP library for future reuse.

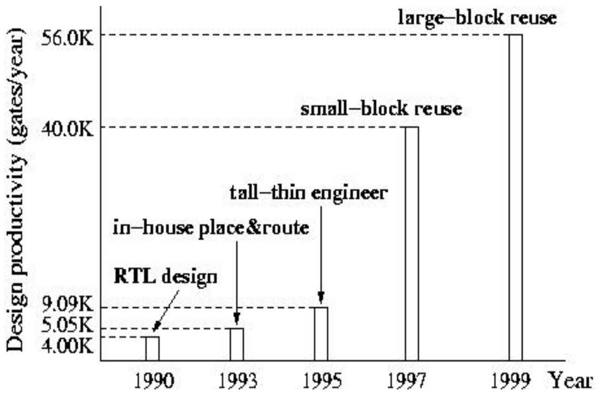


Figure 1. Design technology innovations in the past decade. (data source: *International Technology Roadmap for Semiconductors 2001* [4].)

There exist significant technical barriers to increase design productivity by design reuse. IP protection is among the original six key enabling technologies identified by the Virtual Socket Interface (VSI) Alliance when it was founded in 1996 [1]. It was also cited as a unique and one of the most challenging areas awaiting research breakthroughs.

What makes IP protection a unique challenge is the new reuse-based design environment we just described. It forces engineers to cooperate with others and share their data, expertise, and experience. Design details (including the RTL HDL source codes) are encouraged to be documented and made public to make reuse more convenient. But at the same time, this makes IP piracy and infringement easier than ever. It is estimated that the annual revenue loss in IP infringement in IC industry is in excess of $5 billion. The goals of IP protection include: enabling IP providers to protect their IPs against unauthorized use, protecting all types of design data used to produce and deliver IPs, detecting and tracing the use of IPs [3].

There have been significant advances from both industry and academic towards these goals. Highlights include the VSIA's white paper on IP protection [3] and its physical tagging standard [5] that has now been widely adopted by semiconductor and EDA industry; practices on the protection of web-based collaborate design environment and FPGA design protection by individual companies such as Xilinx (www.xilinx.com) and Altera (www.altera.com); and the numerous protection mechanisms proposed by researchers both from industry and academia on literally every phase of the design procedure[2, 6-16, 18,19,24-26]. Naturally, efforts from industry are on the identify and trace of legal IP usage for the purpose of checking, updating, and royalty reporting among others. Research from academia concentrates on the protection and deterrent of high-tech IP piracy, which is a potentially critical, but not yet real problem in semiconductor and EDA industry. This interesting distinction between industry and academia determines the fact that people will view the same question in different ways and provide different answers within different but relevant context. They are complementary to each other and will collaborate to reach all the goals for IP protection in VLSI design.

In paper, we first provide a comprehensive review of the current state-of-the-art IP protection practice and advances. We then discuss several key challenges, explain how people view these challenges differently from academia and industry. We mention a couple of successful and current efforts from industry. Our main focus is on the directions and practical approaches from academic point of view on how to answer such challenges.

2. State-of-the-Art IP Protections

We restrict our survey to the non-law enforcement IP protection practice and research.

VSIA's physical tagging standard [5] is by far the only official standard and has enjoyed tremendous success in the past few years. It describes a procedure on how to embed information into the graphical design system II (GDSII) file. Such plain text information allows semiconductor manufacturer to tag and track components used (reused) in a design at the fabrication process. This standard is designed to facilitate the information sharing among lawful and honest IP providers and users rather than protect the GDSII file from being misused by adversaries. Note that the standard is open to the public, every foundry can check the information from the GDSII file, and adversary can also easily modify it.

Standard encryption has been mentioned as one way to protect design IPs (including design data) despite the rather expensive decryption process [3]. Recently, several FPGA design tools provide users the encryption choice. Altera's encryption software enables users to encrypt their design and any attempt to compile an encrypted IP will fail if the core is not decrypted properly. Xilinx adds a decryption unit on its latest Virtex-II and Virtex-II Pro FPGA board. Its design tool allows user to select up to six plaintext messages and uses them as the key to encrypt the FPGA configuration bitstream file. The decryption unit will decrypt it before configuring the FPGA board.

On the other hand, protection mechanisms developed from academia serve as the deterrent for IP piracy. Most of these works are under the watermarking framework proposed by Kahng et al. [2,15], which has been applied to various aspects of the VLSI design process, from behavioral and logic synthesis to standard cell place and route algorithms, to FPGA designs [2,7-10,12,15,17,24,25]. There are also several studies on IP fingerprinting techniques [6,14] and methods to recover the embedded signatures [11,16,19,26].

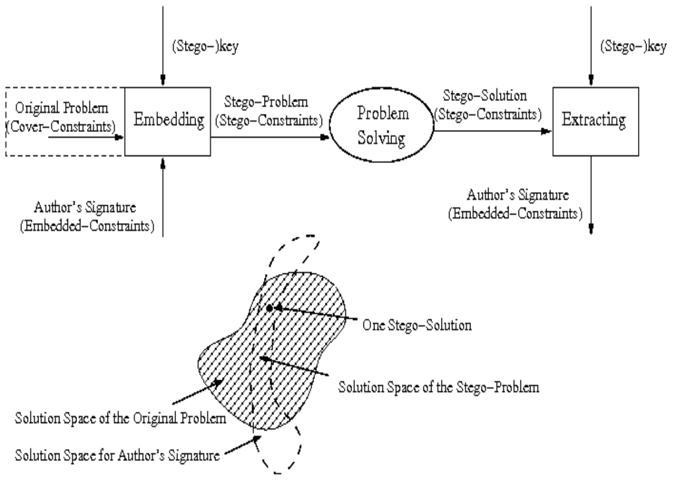


Figure 2. Basic concept of the constraint-based watermarking technique.

The constraint-based watermarking technique [2,15] translates the to-be-embedded signatures into a set of additional constraints during the design and implementation of IP in order to uniquely encode the signature into the IP. Considering IP invention as solving a hard problem where the requirements for the IP serve as constraints, this basic idea can be depicted in Figure 2, where we also show its key concept of cutting solution space.

To hide a signature, the designer first creates another set of constraints using his secret key; these constraints are selected in such a way that they do not conflict with the constraints in the original problem. Then the original and additional constraints are combined to form a more constrained stego-problem. The stego-problem, instead of the original problem, is solved to obtain a stego-solution. Using information hiding terminologies, we refer to the original problem as cover-constraints, and the signature as embedded-constraints. A stego-solution will satisfy both constraints.

To claim the authorship, the author has to demonstrate that the stego-solution carries information based on his signature. The stego-solution unnecessarily satisfies a set of additional constraints that look random until the author regenerates them using his signature together with his secret key. Cryptography functions such as one-way hash and stream cipher will be used to generate the embedded-constraints. There are several approaches proposed to detect the embedded watermark. For several specific instances, Kahng et al. [11] choose signatures selectively and develop fast comparison schemes to detect such signatures. Charbon and Torunoglu [26] discuss copy detection under a design environment that involves IPs from multiply sources that requires IP providers to register their IPs in a trusted agent. Kirovski et al. [19] propose a forensic engineering technique to identify solutions generated by strategically different algorithms.

3. Challenges

There seems to be an interesting and growing rift on the objectives and approaches between industry and academia on IP protection. In industry, design teams are in dire need of techniques and standards that enable them to build yield and risk assessment, checks release numbers, warn them when a problem arises, optimize future library for content definition and enables IP management such as royalty reporting. Being able to protect their IPs from being misused is a nice feature, but unfortunately has not been really considered by many design teams as a necessity. Both the released VSIA physical tagging standard and the soft IP tagging standard [27] look to identify ways to embed design-related plaintext messages into the design to achieve these goals.

In academia, being aware of the existence of powerful reverse engineering tools, researchers are developing techniques at various levels of the design process to discourage any attempt of IP piracy from happening. Their approach is also based on the idea of embedded design-related information into the design. However, such information is encrypted and hidden so the adversary cannot see it with ease.

Nevertheless, both sides have agreed, although again they view the same problem in different ways, on the following new major challenges for IP protection.

3.1 Restrain Design Overhead

Due to its nature of adding constraints, the watermarking-based IP protection techniques inevitably introduce design overhead. As it has been pointed out originally, design overhead should be kept as low as possible [2,11]. Qu and Potkonjak [17] study the random graph coloring problem and conclude that it is possible to keep the overhead at the minimal level while still providing strong protection. However, they also mention that if the number of watermarking constraints is not selected carefully, the overhead can be arbitrarily high. Furthermore, the randomness of the watermarking constraints makes the design overhead hard to predict. For example, Lach et al. experiment the timing and resource overhead for various FPGA watermarking techniques. They report resource overhead from 0.005% to 33.86% and timing overhead from –25.93% to 11.95% [6,9]. Kirovski et al. develop watermarking protocols during the multi-level logic minimization and technology mapping phases. The design overhead goes from negative to 8.12% [8]. Oliveira proposes a technique to watermark sequential circuit designs where the area and delay overhead can be negligible for large designs but are as high as 2747% and 273%, respectively, for even very small designs [10].

Currently, there is no watermarking technique that guarantees the design quality or gives an upper bound on the design overhead. The lack of quality guarantee remains one of the key obstacles for the watermarking-based IP protection techniques to be adopted as an industry standard.

3.2 Protect Soft IP

Soft IPs are delivered in the form of synthesizable HDL codes like Verilog or VHDL programs. Although soft IPs have less predictable performance and much higher protection risks, the emerging trend is that most IP exchange and reuse will be in the form of soft IPs because of the design flexibility they provide [18]. On some occasions, IP provider may also prefer releasing soft IPs to leave customer-dependent optimization process to the users.

From security point of view, protecting soft IPs is a much more challenging task than protecting hard IPs such as GDSII files or fully placed netlist. The flexibility makes soft IPs hard to trace and therefore difficult to prevent IP infringements from happening. IP providers are taking a high risk in releasing their IPs in the soft form before they protect their HDL codes with techniques that are effective, robust, low-complexity, and low-cost. Unfortunately, such techniques or tools are not available and their development is challenging.

Existing constraint-based watermarking techniques for netlist [2,8,15] are applicable to protect gate-level HDL code. Although it is synthesizable, gate-level HDL code is just a plain description of the design and does not help users to understand and thus effectively reuse the design. Synthesizable behavioral-level HDL code, on the other hand, gives a high level description of the system. Its C-like programming syntax and structure makes it friendly for reuse and hence the most valuable asset of soft IPs.

However, existing software protection techniques are not applicable to HDL code protection either. The reason is that these protections are either on the executables or syntax techniques such as removing comments and renaming variables. HDL programs, first of all, do not have any executables or binary codes. Second, they are suggested to have proper documentations and to follow standard coding guidelines for better reusability [18].

3.3 Protect CAD Tool and Algorithm

CAD tools, algorithms, and design environment are yet another type of VLSI design IPs. They are currently being treated and protected as traditional software by mechanisms such as licensing agreements and encryption. Despite the lack of enforcement of licensing agreements and the security holes of encryption protocols, these protections do not provide the ability to detect IP piracy. That is, if a tool or an algorithm is illegally used to generate an IP, one cannot tell from this piece of IP whether it is obtained legally. Because the (dishonest) IP designers can claim that they obtain the IP by other tools or algorithms. Another characteristic on CAD tool and algorithm protection is that piracy normally involves the misuse instead of illegally redistribution of the tool or algorithm.

The only known technique that detects possible CAD tool and algorithm piracy is the forensic engineering approach proposed by Kirovski et al. [19]. It enables the identification of solutions generated by strategically different tools and algorithms. First, statistical data are collected from the solutions generated by a pool of algorithms. Then they study certain problem-dependent properties of the solutions to put the pool of algorithms into clusters. To detect which algorithm has been applied to obtain a given solution, they simply check the given solution for the properties that the algorithm clustering has been performed and claim that the solution is obtained by the algorithm that has the best fit. It has several limitations: first, it is only applicable to distinguish strategically different algorithms. Second, it requires the availability of a pool of candidate algorithms and a large benchmark of problems as well as the computing resource to run each algorithm on each problem instance. Third, characterization of the solutions to cluster the algorithms is not a trivial task.

The collaborated web-based design frameworks [20-22], including IBM's recent launched "on demand collaboration environment to verify chip designs" verification environ-ment [23], make CAD tools more vulnerable than ever to be used misappropriately. Given the important role that CAD tools and algorithms play in the EDA society, particularly in the IP-based design era and the collaborated web-based design environment, the need for effective CAD tools and algorithms protection becomes obvious and urgent.

4. Opportunities

We discuss the possible directions to tackle the above challenges with emphasis on approaches from academia.

4.1 Design Overhead

In the original proposal of watermarking-based IP protection, the pre-processing and post-processing methods are discussed [2,15]. Pre-processing techniques embed watermark before the synthesis tools are applied to solve the (watermarked) problem. In post-processing, the original problem without any watermark is first solved, and then the solution will be altered based on the watermarking constraints. To achieve high robustness of the watermark, the authors argue that it is preferable to have pre-processing and this vision has impacted almost all the follow-up work on IP protection.

As we have already mentioned in the section of challenges, pre-processing techniques add random extra constraints to the initial problem and thus introduces the unpredictable design overhead. Even in the optimization-intensive version [24], no guarantee on the watermarked IP's quality degradation can be provided. However, in the post-processing approach, the best solution from the synthesis tool is known before the watermarking process. It becomes possible to embed watermark in such a way that the overhead can be controlled, if not avoided completely. This is at the cost of reduced robustness because adversary could also make changes on the solution and eventually alter or remove the watermark.

To achieve limited overhead guarantee and high robustness simultaneously, we propose a three-phase watermarking embedded approach. In the first phase, synthesis tool is applied to the original problem to produce the best possible solution. In the second phase, we identify certain non-critical constraints in the problem that will not affect the solution quality dramatically and embed them as the watermark. In the last phase, the synthesis tool is applied again to create a watermarked solution. Comparing to the pre- and post-processing watermarking approaches, this new approach has the following advantages and disadvantages:

1. Phase I conducts the design without any watermarking. This allows us to obtain a design solution with the best possible quality.
2. With the best quality design solution as a reference, we can select constraints to trade solution quality for robustness or other watermarking objectives in phase II. Note that this is impossible for pre-processing because the best design quality is unavailable.
3. Re-synthesis in Phase III makes it again difficult for adversary to temper the watermark. This distinguishes our approach with post-processing.

As one can see, the key step is Phase II where we actually perform the watermarking procedure. We believe that 1) there exist plenty of “redundancies” in the original design constraints, for a given solution, which can accommodate the watermark. We will demonstrate this later in this section by the example of finite state machine state minimization problem. 2) watermarking constraints should be embedded close to the end of the synthesis to reduce the complexity of (re-)synthesis and to increase the predictability of the solution quality change due to watermarking.

As an example, we mention a recent FPGA watermarking approach proposed Jain et al. [25] that guarantees no design overhead in timing and system resource. A design as usual is performed first to get all the timing on all the nets. Then the required signature is mapped to additional timing constraints on carefully selected nets and the *place and route* is redone to generate the FPGA configuration bitstream. Real life FPGA designs demonstrate that all the timing requirements are met with no additional resource. And the watermarked designs have FPGA configuration bitstreams significantly different from the original ones to provide strong proof of authorship (Table 1).

Table 1. Validation of zero overhead and strength of watermark on benchmark FPGA designs [25].

FPGA Designs		Original	Watermarked	Overhead	Bitstrem Difference
DAP (2,503,260 gates)	Resources	1083	1083	**0%**	1.13%
	f_{max} required: 40MHz	√	√	**0%**	
VIDEO (56,253 gates)	Resources	1522	1522	**0%**	2.15%
	f_{max} required: 35MHz	√	√	**0%**	
RISC (6,894 gates)	Resources	746	746	**0%**	5.47%
	f_{max} required: 50MHz	√	√	**0%**	
AddrGen (2,862 gates)	Resources	285	285	**0%**	1.83%
	f_{max} required: 40MHz	√	√	**0%**	

4.2 Soft IP Protection

The IP protection development and working group in VSIA is currently developing a specification/standard for moving soft IPs between companies using a simple tagging or watermarking scheme. This effort is to help honest IP users to exchange IPs. However, it may create a huge security hole for IP protection. In the physical tagging standard, plaintext information on the IP is embedded in the GDSII file and only limited number of semiconductor foundries are capable to misuse such information. An HDL code, on the other hand, can be tweaked easily by any hardware designer. Nevertheless, our belief is that soft IP protection problem, from academic point of view, is solvable.

First, the programming styles and variable naming conventions among other standards suggested by industry experts [18] increase the readability and reusability of the HDL code. Meanwhile, it also limits the power of adversary. With the ongoing effort on soft IP tagging standard, this can only become better once it is adopted. For example, a netlist of the design without documented HDL code makes itself suspicious. Second, technically, it is still possible to hide information into the soft IP. As we will see in the protection of CAD tool and algorithm, every designer has his or her design style that can be traced from the HDL code he or she writes. Moreover, recall that the motivation for soft IP exchange is to make reuse easier and more efficient. Resynthesizing the soft IP with other blocks normally results in designs better than the ones obtained by integrating the hard IP with other blocks. However, certain parts of the soft IP, such as a rather unique memory design, are very unlikely to share resources with other blocks. In such case, providing those parts as hard IP reduces the flexibility of reuse, but it may not affect much of the design quality. Meanwhile, it opens room for hard IP protection techniques. We believe that this combination of hard IP and soft IP will be a promising direction for soft IP protection.

In [28], the authors argue that any good HDL source code watermarking technique should provide (1) strong proof of authorship, (2) low design overhead, (3) survivability from re-synthesis, (4) resilience, and (5) preserve IP's I/O interface. To reach these goals, it is necessary to have the **documentation assumption** and the **verification assumption.** The first requires the designer to document the HDL modules properly and give sufficiently detailed information on each reusable module's input, output, and functionality. It allows, however, designer not to document other details on how each module is implemented. The second requires all HDL design to follow the hierarchical modular fashion and not to mix complicated gate-level HDL code with RT-level description. Based on these assumptions, several techniques have been proposed to embed information into HDL source code. However, these techniques are ad hoc with limited watermark embedding capacity and may have large design overhead.

4.3 Design Tool and Algorithm Protection

Conceptually, it is possible to apply the constraint-based watermarking technique to protect design tool and algorithm during the design and implementation of such tool and algorithm. (To see this, just treat the design tool and algorithm as IPs.) However, quality of the solutions obtained by such protected tool and algorithm cannot be guaranteed. Furthermore, this approach cannot protect existing tools and algorithms without redesign them.

We argue that an effective CAD tool and algorithm protection technique must be able to

- identify with high accuracy that whether the given solution is generated by the target tool or algorithm;
- retain the performance (e.g., CPU and memory requirements) of the tool and algorithm and the quality of the solutions it provides;
- be robust against attempts to remove the protection from the tool and algorithm, or to disguise the solutions obtained by them.

Note that although the design process may not be reversible, CAD tools and algorithms usually leave plenty of traces in the design solution they find. The forensic engineering technique attempts to collect these unintentionally left trace to identify CAD tools and algorithm by sampling over a large set of representative trials. We propose to "intentionally" leave a trace of the tool and algorithm in the design solutions generated by them. This enables us to quickly detect the usage of the protected tool and algorithm with high accuracy. As an example, the following scheme protects a 3SAT solver:

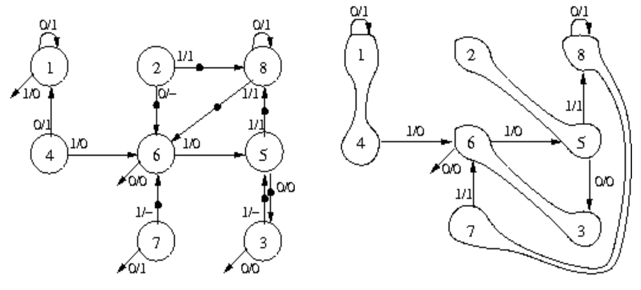
1. select a fixed set of clauses from the 3SAT formula;
2. for each selected clause, $x+y+z$, append $(x'+y'+z')(x+y'+z')(x'+y+z')(x'+y'+z)$ to the formula;
3. solve the new formula to find a solution;

There are seven different ways to make clause $x+y+z$ true. The addition of extra clauses in step 2 enforces that one and only one of the three literals x,y,z can be assigned to be '1'. For any solution obtained by this (protected) solver, the fixed set of clauses (those selected in step 1) will have exactly one literal receiving assignment '1'. This unique feature of the solver can be used to detect whether a given solution is obtained from this solver and thus achieve our goal of CAD tool and algorithm protection.

Yuan et al. [29] propose a birthmarking method for CAD tool and algorithm protection. In their approach, they first use the tool/algorithm to find a design solution with the best possible quality. Then they conduct an additional design step to "birthmark" the design solution by changing it locally without affecting the design quality. Such local changes can be detected later as a proof of the tool/algorithm that has been used during the design. This is similar to the post-processing watermarking technique [2,15]. They use a gate-level timing-driven gate duplication tool as example to illustrate this approach and the extensive experimental results show that such birthmarking technique incur very small overhead.

4.4 Redundant Design Constraints

To end our discussion, we use the finite state machine (FSM) state minimization problem as an example to demonstrate the concept of redundant design constraint [30].



(a) (b)

Figure 3. (a) State transition graph for the FSM. (b) State transition graph for the reduced FSM.

Consider an FSM with eight states, one binary input and one output. Its state transition graph is shown in Figure 3 (a). This FSM can be reduced to one with only four states, and there are two solutions: {{1,4},{2,5},{3,6},{7,8}} and {{1,4,7},{2,5},{3,6},{8}}. Figure 3 (b) is the state transition graph for the first solution.

Surprisingly, if we keep all the output information and only the seven state transitions that have a dot on the arrow in Figure 3 (a), the solution remains the same. This suggests that these conditions are sufficient to obtain the above solutions. The absence of other five state transitions (i.e., replacing the next states of these transitions by don't care will not have any impact. Therefore we call them redundant design constraints.

There are very rich redundant constraints in finite state machine synthesis as indicated by experiments on the standard MCNC sequential circuit design benchmarks [30]. In addition, one can add redundant states, states that are equivalent to other states, and synthesize the non-minimized finite state machines. When part of the redundant states are added, the synthesis solutions have very little overhead, and very often are better solutions, in area and power consumption.

We can leverage this redundancy to hide information by, for example, the following scheme: *keeping the redundant constraint for a bit 1 and deleting it for a bit 0.* Note that we obtain the set of redundant constraints from an existing solution and this modification of the system specification will not change the correctness of this solution. The watermarked design and non-watermarked design will start from this same solution and therefore there will not be any design overhead.

5. Conclusion

VLSI design intellectual property protection has been a hot topic since the late 90’s and is cooled off in the past couple of years. In this paper, we survey the current status of IP protection, analyze the new challenges and discuss the opportunities. Although people from academia and industry have, in some sense, quite different viewpoints of this problem, it is our belief that the collaborated efforts will eventually find ways to protect VLSI design IPs efficiently and effectively.

6. Reference

[1] Virtual Socket Interface Alliance. “Fall Worldwide Member Meeting: A Year of Achievement”, October 1997.

[2] A.B. Kahng, et al. “Watermarking Techniques for Intellectual Property Protection”, *35th ACM/IEEE Design Automation Conference Proceedings*, pp. 776-781, June 1998.

[3] Virtual Socket Interface Alliance. “Intellectual Property Protection White Paper: Schemes, Alternatives and Discussion Version 1.0”, September 2000.

[4] International Technology Roadmap for Semiconductors. http://public.itrs.net/Files/2001ITRS/

[5] Virtual Socket Interface Alliance. “Virtual Component Identification Physical Tagging Standard (IPP 1 1.0)”, 2000.

[6] J. Lach, W.H. Mangione-Smith, and M. Potkonjak. ``FPGA Fingerprinting Techniques for Protecting Intellectual Property," *Proceedings of the IEEE 1998 Custom Integrated Circuits Conference*, pp. 299-302, May 1998.

[7] E. Charbon. ``Hierarchical Watermarking in IC Design," *IEEE 1998 Custom Integrated Circuits Conference*, pp. 295-298, May 1998.

[8] D. Kirovski et al. ``Intellectual Property Protection by Watermarking Combinational Logic Synthesis Solutions", *IEEE/ACM International Conference on Computer Aided Design*, pp. 194-198, November 1998.

[9] J. Lach, W.H.Mangione-Smith and M. Potkonjak, “Robust FPGA Intellectual Property Protection Through Multiple Small Watermarks”, *36th IEEE Conference on Design Automation Conference,* pp. 831-836, June 1999.

[10] A.L. Oliveira. “Robust Techniques for Watermarking Sequential Circuit Designs*", 36th ACM/IEEE Design Automation Conference Proceedings*, pp. 837-842, 1999.

[11] A. B. Kahng et al. “Copy Detection for Intellectual Property Protection of VLSI Design”, *Proc. IEEE/ACM Intl.*

Conference on Computer-Aided Design, pp. 600-604, November, 1999.

[12] I. Hong and M. Potkonjak. ``Behavioral Synthesis Techniques for Intellectual Property Protection", *36th ACM/IEEE Design Automation Conference Proceedings*, pp. 849-854, 1999.

[13] K.W. Yip and T.S. Ng, "Partial-Encryption Technique for Intellectual Property Protection of FPGA-Based Products", *IEEE Transactions on Consumer Electronics*, pp. 183-190, February 2000.

[14] G. Qu and M. Potkonjak. "Fingerprinting Intellectual Property Using Constraint-Addition", *37th ACM/IEEE Design Automation Conference Proceedings*, pp. 587-592, June 2000.

[15] A. B. Kahng, et al., "Constraint-Based Watermarking Techniques for Design IP Protection", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* pp. 1236-1252, October 2001.

[16] G. Qu. "Publicly Detectable Techniques for the Protection of Virtual Components", *38th ACM/IEEE Design Automation Conference Proceedings*, pp. 474-479, June 2001.

[17] G. Qu and M. Potkonjak. "Analysis of Watermarking Techniques for Graph Coloring Problem", *IEEE/ACM International Conference on Computer Aided Design*, pp. 190-193, November 1998.

[18] M. Keating and P. Bricaud. "Reuse Methodology Manual, For System-On-A-Chip Designs," Second Edition, 1999.

[19] D. Kirovski, D. Liu, J.L. Wong, and M. Potkonjak. "Forensic Engineering Techniques for VLSI CAD Tools", *37th ACM/IEEE Design Automation Conference Proceedings*, pp. 581-586, June 2000.

[20] F.L. Chan, M.D. Spiller, and A.R. Newton. "WELD - An Environment for Web-Based Electronic Design", *35th ACM/IEEE Design Automation Conference Proceedings*, pp. 146-151, June 1998.

[21] A. Fin and F. Fummi. "A Web-CAD Methodolgoy for IP-Core Analysis and Simulation", *37th ACM/IEEE Design Automation Conference Proceedings*, pp. 597-600, 2000.

[22] K. Hines and G. Borriello. "A Geographically Distributed Framework for Embedded System Design and Validation", *35th ACM/IEEE Design Automation Conference Proceedings*, pp. 140-145, June 1998.

[23] http://www-1.ibm.com/technology/news/2003/062403ondemand.shtml

[24] G. Qu, J. Wong, and M. Potkonjak. "Optimization-Intensive Watermarking Techniques for Decision Problems", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* Vol. 23, No. 1, pp. 119–127, January 2004.

[25] A. Jain et al. "Zero Overhead Watermarking Technique for FPGA Designs", *13th IEEE /ACM Great Lakes Symposium on VLSI (GLSVLSI'03)*, pp. 147-152, April 2003.

[26] E. Charbon and I. Torunoglu. "Copyright Protection of Designs Based on Multi Source IPs", *IEEE/ACM International Conference on Computer Aided Design*, pp. 591-595, November 1999.

[27] Virtual Socket Interface Alliance. "Soft Intellectual Property (IP) Tagging Standard Version 1.0 (IPP 4 1.0)", August 2004.

[28] L. Yuan, P. Pari, and G. Qu. "Soft IP Protection: Watermarking HDL Source Codes", 6th Information Hiding Workshop (IHW'04), pp. 224–238, LNCS Vol. 3200, Springer-Verlag, May 2004.

[29] L. Yuan, G. Qu, and A. Srivastava. "VLSI CAD Tool Protection by Birthmarking Design Solutions", 15th IEEE /ACM Great Lakes Symposium on VLSI (GLSVLSI'05), pp. 341–344, April 2005.

[30] L. Yuan and G. Qu. "Information Hiding in Finite State Machine", 6th Information Hiding Workshop (IHW'04), pp. 340–354, LNCS Vol. 3200, Springer-Verlag, May 2004.