Zero Overhead Watermarking Technique for FPGA Designs

Adarsh K. Jain, Lin Yuan, Pushkin R. Pari and Gang Qu
Department of Electrical and Computer Engineering, University of Maryland, College Park
{adarsh,yuanl,pushkin,gangqu}@eng.umd.edu

ABSTRACT

FPGAs, because of their re-programmability, are becoming very popular for creating and exchanging VLSI intellectual properties (IPs) in the reuse-based design paradigm. Existing watermarking and fingerprinting techniques successfully embed identification information into FPGA designs to deter IP infringement. However, such methods incur timing and/or resource overhead, unpredictable at times, which causes performance degradation. In this paper, we propose a new FPGA watermarking technique that guarantees zero design overhead.

Our approach consists of two phases. First we design as usual to obtain the best, possible, quality IP. Then we map the required signature to additional timing constraints on carefully selected nets and redo a small portion of the design (e.g. place and route). The FPGA configuration bitstream for the resulting watermarked design will be significantly different from the original design, which provides us with a strong proof of authorship. The watermarking technique has zero design overhead because it is developed to maintain the performance of the design from the first phase. This is demonstrated by applying the proposed technique on several real-life FPGA designs, which range in size from a few thousand to more than two million gates, on Xilinx devices.

Categories and Subject Descriptors

K.5.1 [Legal Aspects of Computing]: Hardware/Software Protection; B.7.1 [Integrated Circuits]: Types and Design Styles—Gate Arrays; J.6 [Computer-aided Engineering]: Computer-aided design; B.m [Miscellaneous] Design management.

General Terms

Legal Aspects, Design, Security, Performance.

Keywords

FPGA, IP protection, zero overhead, configuration bitstream, user constraint file, timing analyzer, place and route, performance.

1. INTRODUCTION

The need for design reuse has been apparent for many years. Today, thousands of designers are creating and exchanging intellectual properties (IPs) on an increasingly large scale. FPGA IPs offer designers flexibility and quick time-to-market and are therefore, a promising alternative to full custom ASICs. Particularly, with the latest series of devices like Virtex-II and Virtex-II Pro [19] from Xilinx [25], FPGAs have started taking a significant share of the integrated circuit market [18]. For example, some devices in the Virtex-II Pro series have up to four embedded IBM Power PCs, making it feasible to use FPGA for large scale computing. On the other hand, it becomes impractical to design such FPGA devices with millions of gates from scratch. Naturally, reuse-based design methods are very useful for FPGA designs. Already, companies like Xilinx and Altera are offering reusable IPs for use with their tools and devices. Xilinx also provides its *FPGA Reuse Methodology Manual* [21].

As design reuse becomes common, design exchange will take place between vendors, which requires protection techniques for the safe exchange of designs at various levels of abstraction. Kahng et al. [4] proposed the first constraint-based watermarking technique in which the author's signature is mapped into a set of constraints that can be independently satisfied for a particular solution. These additional constraints are embedded as watermark into the original design. Kahng et al.'s technique can be used to protect designs from the high-level synthesis level to the physical level, as well as FPGA designs [8]. They treat the CAD tool as a black box and add constraints in either the pre-processing or the post-processing stage. Pre-processing is preferred as embedding watermark in the post-processing stage will make it vulnerable to being removed by reverse engineering. However, the pre-processing approach introduces additional constraints into the design and will lead to, in general, an inferior design.

A typical FPGA design flow includes a number of stages including design entry, synthesis, place and route and finally, configuration of the device. Most commercially available CAD tools provide user-friendly interface to control individual design stages. At each stage, a number of choices are available to the designer: optimize design for area or for speed, allow register duplication or not, and so on. Our technique leverages this fact and embeds watermark at the *place and route* stage by manipulating delays on certain selected nets. What distinguishes our approach from Kahng et al's technique is that we embed watermark after designing without watermark first, and, that we then repeat a part of the design. In this way, we have better control over design overhead and we can make the watermark more robust.

Many FPGA IPs are delivered in the form of configuration files [16,17]. The configuration file is a very large bitstream file (in the order of tens of megabytes) loaded in the FPGA's Static-Random-Access-Memory (SRAM). It defines the functionality of the FPGA. A change in even a single design constraint, even at late design stages, results in a configuration file that differs from the original file by thousands of bits. On the other hand, reverse

GLSVLSI'03, April 28-29, 2003, Washington, DC, USA.

engineering the bitstream file, i.e., determining a set of design constraints to re-generate a given bitstream file, is virtually impossible [17]. This feature makes our FPGA watermarking technique secure and robust because the IP designer is the only one who can reproduce the same bitstream file, from a set of design constraints, which has the designer's signature embedded in it.

The rest of the paper is organized as follows. Section 2 surveys related work on VLSI design IP protection. In Section 3, we briefly introduce the Xilinx FPGA design platform and use it as an example to explain our zero overhead FPGA watermarking approach. Section 4 reports the experimental results of applying our watermarking technique to several real-world FPGA designs. We conclude in Section 5.

2. RELATED WORK

We restrict our survey to VLSI design IP protection and particularly to the protection of FPGA designs. Other related areas include steganography, information hiding, encryption, multimedia content protection, software obfuscation, network security, privacy protection, and so on. (We refer interested readers to [1,3,4,6,13] and their references).

According to the IP protection white paper released by the Virtual Socket Interface Alliance [20], there are three approaches to secure an IP: *deterrent approaches* like patents, copyrights, and trade secrets; *protection* via licensing agreements or encryption; *detection mechanisms* such as physical tagging, digital watermarking and fingerprinting. Among them the detection mechanism is the one that has attracted a lot of attention in recent years. As we have already mentioned in the introduction section, Kahng et al. [4] first established the desiderata for constraint-based watermarking techniques for the protection of VLSI design IPs. The protection is achieved by tracing unauthorized reuse and by making it as difficult as re-designing the IP from scratch. This approach 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 [2,4,5,7,10,11]. There are also several studies on IP fingerprinting techniques [1,7,12] and methods to recover the embedded signatures [3,6,13].

To the best of the authors' knowledge, there are three pieces of work reported on FPGA design protection. Lach et al. [8] proposed a FPGA watermarking technique by using post-processing constraints. The essence of their approach is to encode the signature bits and embed them into the unused look-up tables (LUTs) such that they do not affect the original design and then reroute the design around these LUTs. The disadvantage of this approach is that the watermark is not embedded as a functional part of the design; given enough information, the watermark can be removed without affecting the design functionality. Further, it is not easy to automate the entire watermarking process.

The same authors later [7] improved their watermark's robustness by embedding multiple small watermarks, instead of a large watermark, all over the design. However, there are two problems associated with this approach. First, extra resources will be used, which may affect the overall performance and power consumption of the design, although no such data was reported in the paper. Second, the constraints imposed on the placement of the embedded watermark may also contribute to timing overhead. No guarantees on such overhead could be provided due to the watermark's pre-processing nature. This is reflected in the paper by the rather random timing overhead of up to 11%.

Another interesting approach is based on encrypting the bitstream file. Yip et al. [16] proposed a partial encryption scheme, in which the configuration bitstream is partially encrypted and then loaded on to a separate RAM built into the FPGA. This requires a decryption unit on the FPGA to read and decrypt the encrypted bitstream from the above special-purpose RAM, and then load the decrypted bitstream into the main configuration RAM. The security of this technique relies on the fact that the bitstream file is hard to reverse engineer. A similar method of bitstream encryption is being supported (though in a slightly different way) in some new devices such as the Virtex-II and Virtex-II Pro from Xilinx. However, this requires additional hardware on the FPGA, and therefore is not generic and the additional decryption unit may cause some overhead in terms of power consumption.

3. WATERMARKING TECHNIQUE

Due to the increasing complexity of FPGA devices and time-to-market pressures, designers rarely build the entire design from scratch. Large FPGA vendors provide FPGA design tools to support their newest devices and facilitate the design process. To protect FPGA designs, we leverage some inherent features of the design tools to embed watermark in the design. There are two advantages of this approach: first, the signature becomes, in a way, a functional part of the design and, second, since no constraints are added to the design specification, we avoid possible overhead as far as area and timing are concerned. In this paper, we build our watermarking technique on the Xilinx ISE 5.1 FPGA design platform. Specifically, we use the Timing Analyzer Tool (available internally with the ISE) to select nets from the design to add our watermark. Usually, all FPGA development tools provide similar interface and so our technique, though we worked on the Xilinx ISE 5.1, can be easily applied to other tools as well.

3.1 Basics on Xilinx ISE 5.1

We first briefly introduce the Xilinx Integrated Software Environment (ISE) that we use to illustrate our zero overhead watermarking technique. ISE includes various design entry, synthesis and implementation tools enabling designers to verify and analyze the design at every stage.

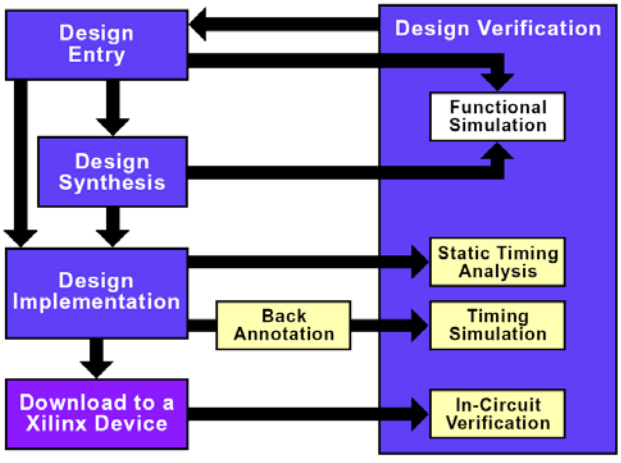

Figure 1. Design Flow in Xilinx ISE [25].

Figure 1 depicts a typical FPGA design flow [25] in ISE 5.1. The design can usually be entered in HDL or as a schematic or as an FSM. A suitable FPGA is then selected. After syntax checking and compilation, the design is ready for synthesis, which can be done by the Xilinx synthesis tools, which come as a part of the ISE, or by third party tools such as Synplify, which can be integrated with the ISE. The synthesized netlist is then sent to the Place and Route tools to implement the design on the given device. The implementation tools report various statistics about the design like resource utilization, timing, area etc. The Timing Analyzer lets us verify that the delay along a given path or paths meets the specified timing requirements. This can be done once the design has been placed and routed. The tool gives timing details of all the nets of the design such as those between flip-flops, RAMs and pins etc.

Various constraints can be defined and put on the design to meet the specified timing or area requirements. Designers can specify their constraints in a file called the User Constraints File, which can be created and modified via the Constraints Editor (another internally integrated tool). Once the design meets all the specs, it is ready to be put on the device. The configuration bitstream files are then generated (again using internal Xilinx tools in ISE). This configuration file can then be loaded into the on-chip RAM of the FPGA for configuration of the various switches of the device.

3.2 Watermarking Approach

One can control the delay on each net in the design by adding the required timing constraint on that net in the *user constraints file* in ISE. This file is integrated with the design during implementation and eventually it affects the place and route result. Delays along critical paths are usually controlled in order to meet the system performance requirement. The timing on other paths is more flexible and a slight change will not affect the design's overall performance.

Consider the nets between flip-flops in synchronous circuits. The delay on such nets must be less than the desired system clock period. For example, if we need a synchronous design to run at 100MHz, then the delay on the net cannot be longer than 10 ns. If the delay is 7.2 ns, then enforcing it to be 7.1 ns or 7.0 ns in the user constraints file will have little impact on the performance of the circuit. (We assume, of course, that the design is not very tightly constrained already such that a new implementation cannot be found and that all the delays on critical paths are constrained and will not be affected by the above change on the delay of a non-critical net).

However, different delays on paths will force the place and route result to be different. This in turn will require a different set of switches on the FPGA to be opened or closed. Therefore, the configuration bitstream generated by ISE, which programs these switches, will be different. In fact, we will show in Section 4 that changing the delay on only one net will affect a significant number of configuration bits. This provides us with a rather unique bitstream file as a strong proof of authorship.

Figure 2 outlines the zero overhead watermarking approach**.** We first synthesize and implement the design as usual, without any watermark. We then use the timing analyzer provided by ISE to obtain the static timing analysis data on all the nets. We can use the filter settings to keep only the paths between flip-flops. Next, we select a certain number of nets, corresponding to the length of the signature bitstream. Then we use our watermark embedding scheme to convert each signature bit to a constraint on the delay of one of the selected nets. This step can be done by ISE's Timing Constraints Editor. We will show an example watermark-embedding scheme below. Finally, we replace the timing constraints on the selected nets in the user constraint file by these modified delays and redo the place and route phase of the design.

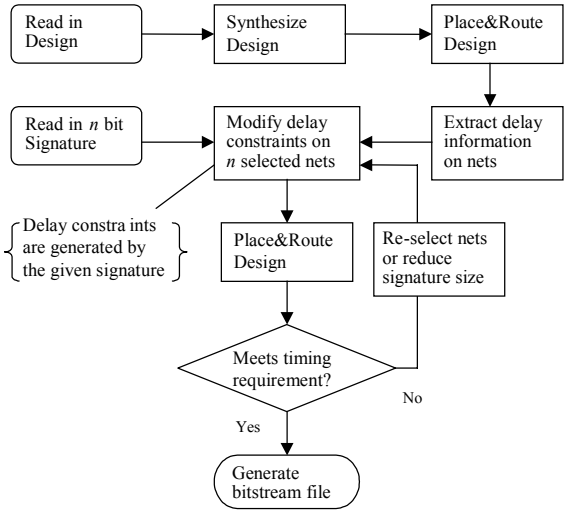

Figure 2. Watermarking Process.

A simple watermark-embedding scheme can be defined as follows: replacing the last digit of the delay by the watermark bit. That is, truncating the digit to embed a bit '0' and replacing it by '1' to embed a bit '1'. We now show how to embed the letter 'M', 01001101 in ASCII with the most significant bit as the parity bit, into a RISC core implemented on an FPGA. Table 1 lists the eight selected nets and their original delays from the timing report. The last column shows the new delays we will use to implement the watermarked RISC core.

Table 1. Example of modifying timing constraints corresponding to signature.

Source FF name	Destination FF name	Original Delay (ns)	Watermark Bit	Constraint Delay (ns)
inst_reg_0	aluinp1_reg_3	17.758	0	17.750
inst_reg_6	aluinp1_reg_3	17.755	1	17.751
inst_reg_1	aluinp1_reg_3	17.733	0	17.730
inst_reg_3	aluinp1_reg_3	17.651	0	17.650
inst_reg_6	aluinp1_reg_1	17.374	1	17.371
inst_reg_1	aluinp1_reg_1	17.352	1	17.351
inst_reg_5	aluinp1_reg_3	17.312	0	17.310
inst_reg_0	aluinp1_reg_5	17.066	1	17.061

The chance for the watermarked design to fail in implementation is extremely low due to the following reasons. First, it is recommended for an FPGA design to use 75-80% of the resources; second, we select the nets off critical paths and we only

reduce the delay by less than a hundredth of a nanosecond (in the worst case, truncating a '9' in the last digit). These indicate that we will not require an FPGA device of larger size to implement the watermarked design. And the timing requirement is automatically satisfied from the way we embed watermark. Therefore, our approach has zero overhead.

Although we have mentioned that the change of a single constraint may result in a significantly different configuration bitstream file, it is still interesting to see how much information we can hide into the FPGA design in this way. This is also important for FPGA fingerprinting when we need to embed distinct signatures into the same design to create IPs with identical functionality but different implementations. Note that in our approach one selected net can hold one bit information from the signature file. Most real world synchronous designs have a large number of flip-flops and so one can easily embed watermarks of several kilobits into the design by our method.

Finally, we mention that the signature generation process can make use of some sophisticated encryption systems. Designers can encrypt their company's name or other identification information using standard encryption systems (such as MD5). The encrypted binary bits are then embedded as signature into FPGA designs using the method explained above. This makes the watermark more secure like in other IP protection methods [4,7].

3.3 Analysis of the Approach

Our method requires no extra hardware and thus can be easily implemented on existing devices. The large number of nets in the design provides us with sufficient room to embed long watermarks. Also, it can be applied to existing synthesized designs as well, as we embed the watermark at the post synthesis stage. The design only needs to be re-implemented, which cannot be avoided anyway.

The proof of authorship in our method relies on the uniqueness of configuration bitstream files for a given set of constraints. Given different timing constraints, the CAD tool will always generate a different bitstream file. From the information available in the documentation provided by Xilinx on its website [25] and also from our discussion with their engineers [17], it is believed that the probability of generating the same bitstream file with two different sets of timing constraints is virtually zero. Furthermore, since the bitstream file is usually large in size (up to a few megabytes), and since its specifications are not known, trying to reverse engineer the design by inspecting the bitstream file will be extremely time consuming and infeasible. Statistics have shown that the first 80% of the configuration information can be determined relatively easily by inspection, but the next 16% is much more difficult [7].

The overhead in our technique in terms of timing and resource utilization will be zero. Since we only play with the net delays, the technique does not use any extra resources and therefore has virtually no impact on the power consumption or resource utilization of the design. So, independent of the signature size, the resource utilization of the FPGA will remain unchanged. Also, the nets used to embed the watermark are selected in such a way that none of them is a part of the critical path. To make sure that the design meets its performance requirements, suitable constraints can be put on the nets, which lie on the critical path. In this way we can ensure that changes caused in the design as a result of watermarking will not affect the critical path. The only possible overhead is the amount of extra time the tool may take to place and route the watermarked design because of the timing constraints enforced on certain nets. However, this, generally, does not take much time unless the resource utilization percentage for the design is extremely high (above 80-85 %), which is generally not recommended for FPGAs.

The proposed technique is generic in the sense that it can be easily integrated with the CAD tool and the whole process of embedding a watermark, we believe, can be fully automated.

4. EXPERIMENTAL RESULTS

We applied our technique to several real-world FPGA designs: a Data Acquisition Path (DAP) [22], a RISC core [24], a Video Encoder [24] and an Address Generator [23]. We implemented these designs on the Xilinx Virtex-II family of devices. We chose for each design, a suitable device according to its size. The Data Acquisition Path is the largest one with more than two million gates and was implemented on XC2V3000. The Video Encoder is

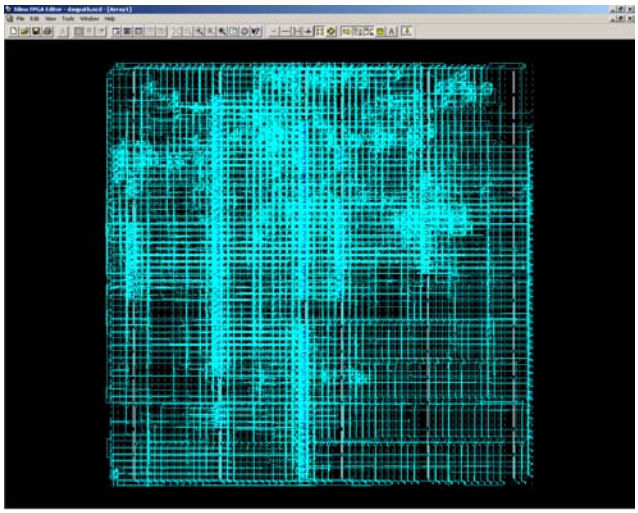

Figure 4. Routed original DAP design.

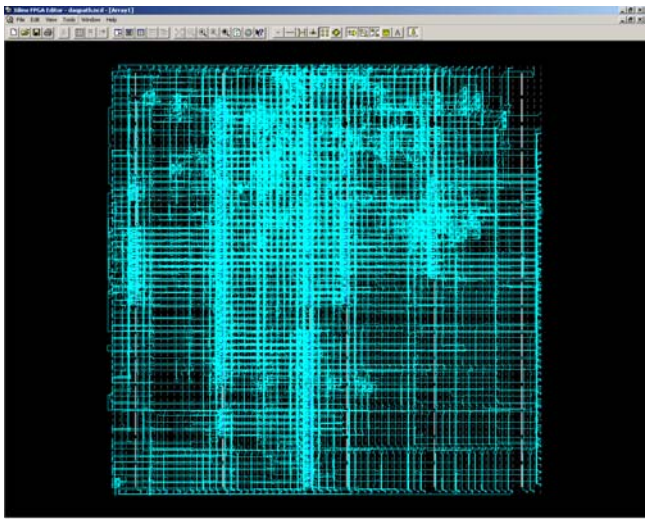

Figure 5. Routed DAP design with 16-bit signature.

of moderate size (50 thousand gates) and was therefore implemented on the relatively smaller XC2V500. The RISC core and Address Generator were implemented on the small XC2V80 due to their small sizes (five thousand and two thousand gates respectively). For all our work, we used the Xilinx ISE 5.1

We implemented these designs following the normal FPGA design flow in Xilinx ISE and generated the bitstream configuration files. We applied our watermarking technique and re-implemented the design to generate the watermarked bitstream file. We then computed, for each design, the Hamming distance between the bitstream files generated with and without watermark. To measure the impact of the watermarking constraints on the bitstream file, we first embedded a one bit watermark in the DAP design. We did this ten times and each time we added the constraint on a different net. We observed, on an average, a 0.7% difference in the bitstreams. This design was implemented on the large XC2V3000 FPGA, whose bitstream contains more than 10 million bits. Thus a 0.7% difference resulted in more than 70,000 bits being changed due to the introduction of a constraint on a single net. If we embed 16 bits, the difference becomes 1.13%, that is, 119,512 bits. This noticeable difference in the bitstream is sufficient to prove authorship. Furthermore, it is virtually impossible to distinguish between bits coming from the watermark or those coming from the original design. This makes the task of reverse engineering very difficult.

Figures 4 and 5 show the fully routed DAP design, before and after watermarking. As can be observed, just by looking at the implementation of the design, it is not possible to conclude that the design is watermarked or where in the design is the signature embedded.

Table 2. Validation of zero overhead and strength of watermark on benchmark FPGA designs.

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%**	

To demonstrate that our technique has zero overhead, we embed a 16-bit signature '0100110101000100' (ASCII code of 'MD' in binary) in the four selected designs. We refer to the Place & Route Report and the Timing Analysis results from the tool to report the results. Table 2 reports the statistics for resource utilization, frequency requirements and bitstream differences. We can see that there is no overhead on resource utilization, as expected. The overhead in timing is also zero, because both before and after watermarking, the delays between flip-flops are less than the clock period, that is, they satisfy the system frequency requirement. Our results show that the watermarked design provides the same quality as the un-watermarked design to the customer. As the only cost of our technique, the designers need extra time to re-implement, i.e., re-place and re-route the design. We report in Table 3 the amount of CPU time required to place and route the watermarked design. As we can see, there are increases in total implementation times ranging from 10 to 46 seconds (which includes increases in placement times from 4 to 14 seconds and increases in routing times from 6 to 32 seconds) depending on the design size. These increases are due to the extra effort that may be required on part of the place and route tool to meet the constraints on the selected nets. However, this is not really an overhead from the point of view of the end user of the design as the design is still able to run at its required frequency, utilizing the same number of resources. Another aspect of our technique is that it uses information, which has already been generated by the tool. Thus, integrating our technique with the tool will not be very difficult, as no new information needs to be extracted from the design.

Table 3. CPU time for place and route(in seconds).

FPGA Designs		CPU Time (second)		
		Placement	Routing	Total
DAP	Original	51	78	129
	Watermarked	65	110	175
	Extra	14	32	46
VIDEO	Original	9	14	23
	Watermarked	16	24	40
	Extra	7	10	17
RISC	Original	3	5	8
	Watermarked	7	10	17
	Extra	4	5	9
Address Generator	Original	2	2	4
	Watermarked	6	8	14
	Extra	4	6	10

5. CONCLUSIONS

In this paper we propose a new watermarking technique to protect FPGA designs. The main feature of this technique is that, unlike existing techniques, it will not introduce any area or timing overhead. We achieve this by designing without any watermark and then re-implementing the design to embed the watermark. This new watermarking method has two advantages. First, watermark becomes an inherent part of the design after re-implementation and is thus resilient and robust. Second, our approach is developed in such a way that it maintains the original design quality even after embedding the watermark.

We demonstrate these on several real-world FPGA designs on the Xilinx development platform, while our approach can be easily used with other FPGA development tools. We embed the watermark in the *place and route* phase of the design cycle but it is also possible to watermark at other design stages or even across multiple stages. We are currently working on extending our technique to FPGA fingerprinting and the protection of ASIC designs.

6. ACKNOWLEDGEMENTS

Our sincere thanks to Dr. Tullio Grassi of the High Energy Group in the Physics Department, University of Maryland, for his valuable suggestions and for his kindness to let us use one of the designs from his group for this work.

We would also like to thank Professor John Lach, Department of Electrical and Computer Engineering, University of Virginia, for sharing with us some insights on his early works on this topic.

7. REFERENCES

[1] A.E. Caldwell, H. Choi, A.B. Kahng, S. Mantik, M. Potkonjak, G. Qu, J.L. Wong. "Effective Iterative Techniques for Fingerprinting Design IP", *36th ACM/IEEE Design Automation Conference Proceedings,* pp. 843-848, June 1999.

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

[3] A. B. Kahng, D. Kirovski, S. Mantik, M. Potkonjak, and J. L. Wong, "Copy Detection for Intellectual Property Protection of VLSI Design", *Proc. IEEE/ACM Intl. Conference on Computer-Aided Design*, pp. 600-604, November, 1999.

[4] A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. Potkonjak, 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.

[5] D. Kirovski, Y. Hwang, M. Potkonjak, and J. Cong. "Intellectual Property Protection by Watermarking Combinational Logic Synthesis Solutions", *IEEE/ACM International Conference on Computer Aided Design*, pp.194-198, November 1998.

[6] D. Kirovski, D. Liu, J.L. Wong, M. Potkonjak, "Forensic engineering techniques for VLSI CAD tools", *IEEE/ACM Design Automation Conference,* pp. 580-586, June 2000.

[7] 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.

[8] J. Lach, W. H. Mangione-Smith, M. Potkonjak, "Signature Hiding Techniques for FPGA Intellectual Property Protection", *IEEE/ACM International Conference on Computer Aided Design*, pp. 186-191, November 1998.

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

[10] M. Ohlrich, C. Ebeling, E. Ginting and L. Sather, "SubGemini: Identifying Subcircuits Using a Fast Subgraph Isomorphism Algorithm", *Proceedings of the 30th IEEE/ACM Design Automation Conference*, pp. 31-37, June 1993.

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

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

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

[14] J. Smith and B. Comiskey, "Modulation and Information Hiding in Images", *First International Workshop on Information Hidin*g, 1996.

[15] G. A. Spanos and T. B. Maples, "Performance Study of a Selective Encryption Scheme for the Security of Networked, Real-Time Video", *International Conference on Computer Communications and Networks*, 1995.

[16] 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.

[17] Personal Communication with Chris Mead, Xilinx Inc.

[18] Synopsys White Paper on FPGA solutions.

[19] Virtex™-II Platform FPGA Data Sheet.

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

[21] Xilinx FPGA Reuse Methodology Manual, 2nd Edition.

[22] High Energy Group, Physics Department, University of Maryland.

[23] Benchmark Suite for Placement, CAD/VLSI Lab, National Tsinghua University. http://nthucad.cs.nthu.edu.tw/~ycchou/benchmark.

[24] www.opencores.org.

[25] www.xilinx.com.