Fragile IP Watermarking Techniques

Amr T. Abdel-Hamid*, and Sofiène Tahar§

*Faculty of Information and Engineering Technology, German University in Cairo, Cairo, Egypt
Email: amr.talaat@guc.edu.eg

§Electrical and Computer Engineering Department, Concordia University, Montreal, Canada
Email: tahar@ece.concordia.ca

***Abstract*— Intellectual property (IP) blocks reuse is essential for facilitating the design process of system-on-a-chip. Sharing IP designs still poses significant high security risks not only to copyright but also to design authenticity. Intruders, or even competitors, can make design changes to IPs, which can lead to the loss of the owner's credibility. In this paper, we are trying to solve such challenge by proposing a novel fragile IP watermarking technique. The proposed technique protects hardware designs from alteration or any modifications that might occur to the design. The approach utilizes existing transitions in a finite state machine (FSM) component of an IP and does not result on any overhead to the IP design. Finally, we implemented the algorithm proposed and tested it.**

I. Introduction

Fast advancing IC (integrated circuit) processing technologies have enabled the integration of full systems on 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 [3], 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 platforms for reducing development time and efforts. Reusable virtual components or Intellectual Property blocks (IPs) [3] are most effective when it comes to reducing cost and development time of SOC designs.

IP blocks are delivered in three main flavors depending on price, applications, and contracts between companies. The Virtual Socket Interface (VSI) architecture document [18] describes such levels as: Soft, Firm and Hard IPs. ***Soft IPs*** 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 (i.e., timing, area, power). Soft IPs typically have increased risks because RTL (register transfer level) source code is required by the integrator and it can be easily modified. ***Firm IPs*** have been optimized in structure, 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. Risks are equivalent to those of Soft if RTL is included, and are less if it is not included. ***Hard IPs*** 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 they are extremely difficult to modify in such form.

Sharing IP designs poses significant security risks. IPs need time and effort to be designed and verified and they are considered a big investment to the companies developing them. Security risks that are facing IP designs can be classified as either *Copyright Violation* or *Intentional Tampering*. An intruder accesses the IP and resells it without getting the permission to do so from the copyright owner is what is named *Copyright Violation*. On the other hand, *Intentional Tampering* is when an Intruder modifies the IP content in order to insert features or modify the IP behavior. Developers and owners of IP designs want assurances that their content will not be illegally redistributed by consumers (*Copyright Violation*), and consumers want assurance that the content they buy is authentic (*Intentional Tampering*).

As a *Detection* approach, watermarking solves both *Copyright Violation* and *Intentional Tampering*. Watermarking techniques can be divided into two sub-domains: Robust Watermarking that has the additional requirement of robustness against possible attacks, and fragile watermarking that is extremely sensitive to design changes. *Robust Watermarking* techniques have the property of being infeasible to be removed or make them useless without destroying the object (IP) at the same time. This makes it adequate for *Copyright Violation* because watermarking (tagging) the IP design will give the owners the ability to track their design. *fragile watermark* is destroyed as soon as the IP is modified or altered. This makes it adequate to prove design authenticity because any intentional tampering in the design will be detected directly. Modifying an IP design carries another sensitive risk to the developer, as this modification can be made by a competitor or an ex-employee that tries to sabotage the developed IP. Also, because of global economic pressures, fabrication of advanced integrated circuits is migrating to foreign foundries.

978-0-7695-3166-3/08 $25.00 © 2008 IEEE
DOI 10.1109/AHS.2008.73

Authorized licensed use limited to: CONCORDIA UNIVERSITY LIBRARIES. Downloaded on January 7, 2009 at 15:11 from IEEE Xplore. Restrictions apply.

These trends have raised concerns regarding many sensitive systems (such as military, and medical systems) that rely on high performance microchips and potential vulnerabilities to these systems caused by malicious manipulation of hardware that might render these vital systems inoperable at some future time. Clearly, a small change in any design can introduce a bug to the whole system. IP developers should have the ability to detect any changes the customer had made to his/her IP. The IP developer investment would be in risk if he/she could not prove that such bug was introduced by another party. On the other hand, customers need to ensure the authenticity of the IP they will use. The only way to do this now is comparing their original IP as well as the design under test. This can be effective if both designs are still in the same level of abstraction, but it is mostly impossible if the designs are at different levels.

Finite state machines (FSMs) are a basic part of hardware designs, they are the transformation between inputs and outputs of the design and are present in the control part of any IP. FSMs are related directly to input/output of the design, and can be detected on mostly all lower abstraction levels. Finally, FSMs can be presented in many different ways, such as state transition graphs (STG). Behavioral level (FSMs) watermarking has several advantages over other watermarking approaches. Watermarking high levels in the design cycle will be automatically inherited in all the lower design levels. This approach was commonly used for copyright protection technique, such as [17],[13], and [2]. In this paper, we propose a novel fragile IP watermarking technique based on the ideas of using coinciding transitions of FSMs for copyright protection [2]. To the best of our knowledge, this is the *first* time a fragile IP watermarking technique is proposed.

The rest of the paper is organized as follows: Section II gives a brief description of related work on IP watermarking. Section III presents the proposed watermarking framework as well as the required insertion and extraction algorithms. Section IV presents a prototype implementation of the algorithm including different experimental results. Finally, Section V concludes the paper.

II. IP Digital Watermarking Techniques

A. Digital Watermarking

History has provided countless situations whereby information has had to traverse hostile or enemy territory to reach its destination undetected. Khan [8] mentioned many examples about hiding data on another media as a cover throughout history. For example, 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. Perhaps one of the famous examples of copyright protection is the clear signature of most of the famous painters on their paintings that is extremely hard to be copied or imitated.

Watermarking is a sub-domain of *steganography*, *Stegano* means hidden or covered in Latin, which gives the meaning "covered writing". In their survey about data hiding techniques, Petitcolas *et al.* [14] 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 utilizes the secrecy and undetectability of steganography to transfer secret data, used mainly for espionage applications. Second, *intellectual property* protection applications, where the watermark is mainly used to convey the information about content ownership and intellectual property rights.

Finally, *content verification* applications (*authentication*), where a fragile watermark is introduced to secure the contents integrity. A *fragile watermark* is destroyed as soon as the object is modified or altered. This can be used to prove any intentional tampering in the design.

Copyright marking (widely known as *watermarking* or *fingerprinting*), as opposed to steganography, has the additional requirement of robustness against possible attacks. Robust watermarking has 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 [14]. On the other hand, *fingerprinting* [14] is like a serial number that enables the intellectual property owner to identify which customer broke the license agreement.

The watermarking process is divided in 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 watermarking algorithms are usually based on one or more of the four basic techniques [14]: 1) Security through obscurity, where the designer tries to cover the way he/she used to embed the watermark; 2) Camouflage, or making the hidden data expensive to look for; 3) Hiding the location of the embedded information; 4) Spreading the hidden information. Also there are many other techniques that depend on the environment of the stego-object.

B. IP Watermarking: Related Work

There are many IP watermarking or fingerprinting techniques available in the open literature. Kahng *el al.* [9] proposed and experimented a *constraint-based* IP watermarking as one of the leading static approaches. The approach is a generic algorithm that can be used at different levels of the design flow. It is based on available tools used mainly to solve NP-hard problems. This was done by adding a set of well defined extra constraints that generate a watermarked solution for

Authorized licensed use limited to: CONCORDIA UNIVERSITY LIBRARIES. Downloaded on January 7, 2009 at 15:11 from IEEE Xplore. Restrictions apply.

the problem. The approach was tested and applied to different levels of the IP design. At the system level for instance, it was used for the watermarking of memory graph coloring solutions by Hong *el al.* [7]. At lower design levels, the approach was used even more effectively in routing [12], placement, and floor planning [9]. The main advantage of the approach is its real low overhead on the design cycle, as the NP-hard problem will be solved anyway. The approach however, has two main drawbacks. First, tracking the watermark is not that easy if the design is resold at other abstraction levels. Also, imposing extra-constraints on a design procedure might not be as successful as thought by the authors. Finally, Le *et al.* [10] showed that several constraint-based watermarking schemes can be broken easier than previously thought.

At the behavioral level, there are mainly three approaches developed for robust watermarking Oliveira [13], Torunoglu *et al.* [17], and Abdel-Hamid *et al.* [2]. The three approaches are based on adding new input/output sequences to the FSM representation of the design. Their main advantage is the ability to detect the presence of the watermark at lower design levels. The algorithm in [17] is mainly based on extracting the unused transitions in a 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 [13] tries to manipulate implicitly the STG, where the user changes the design to include the watermark as a specific property, which is rare in non-modified STGs. This also cannot be done mechanically as it might affect the design functionality.

The approach in [2] is based on utilizing both existing and unused transitions in the FSM to embed the signature. Using existing transitions provides a supraliminal channel[1] as it would give more strength to the system against different attacks. This helps balancing between adding enough data to identify the owner and the design overhead (area, power, delay) this data may introduce. On the other hand, the unused transitions guarantee the uniqueness of the watermark, and ensure the addition of the desired watermarking sequence.

The robust watermarking techniques introduced above protect IP designs form any copyright violation. For a complete protection model against piracy, a protection against intentional tampering is still needed. Fragile watermarking is used in the multimedia domain to detect any intentional tampering that might occur in the watermarked object. In contrast to the robust watermark, the fragile watermark tries to map any small design change, hence ensuring the authenticity of the design. In this work, we are extending the approach developed in [2] to provide a complete protection model for IP designs.

III. IP Fragile Watermarking Technique

A. Fragile Watermarking Framework

Figure 1 shows the proposed framework for fragile watermarking. The framework is mainly divided into insertion and extraction phases. To generate the data needed for watermarking, the set of all outputs of the design should be generated. This set can be generated using for example the transition tour algorithm [6]. Any other technique that generates the whole set of the FSM output can be used as well. The designer then hashes these output data to minimize its size using any hashing technique (such as MD5 [15]). Finally, using the insertion algorithm presented in the next subsection, the designer can automatically add the data to the IP design under investigation. Such insertion process results in generating an extraction sequence that is considered as the watermarking key that will be used for the extraction process.

The designer can extract the design digest using the key provided, and compares it to the one generated from the implemented design whenever he/she needs to check for design changes according to the extraction algorithm shown below. This comparison will simply show the exact location of all design alterations. This location gets more accurate when the window we use to divide the outputs gets smaller.

The fragile watermark is robust in the sense that it cannot be extracted or deleted from the design without the knowledge of the key, but it is fragile in the sense of detecting any changes that might happen to the design. This operation has a main advantage over comparing all design outputs, as this operation needs less time, and it does not need the original design to be available. Finally, it is not effected by the abstraction level of the design.

B. Fragile Watermarking Insertion Algorithm

In [2] a robust watermarking insertion algorithm, named the *output mapping algorithm* has been developed. It attempts to coincide a part of the watermark on the STG transitions to increase the watermark robustness. This is done by searching different outputs for each visited state in the STG, and comparing it to a part of the generated signature in order to map this signature on the system outputs.

We propose a fragile IP watermarking algorithm for FSMs that can be considered a derivative of the above algorithm. Figure 2 illustrates an example for the proposed fragile watermarking algorithm using the signature given at the bottom of the figure. Starting from state $q0$, we check if the first bit in the digest (0) can be mapped with any output bit in the two available transitions. This bit can be clearly mapped on both transitions. We go and try to map the second bit (0), this bit can be mapped on the transition connecting $q0$ with $q1$. Thus this transition is chosen as it enables us to embed more bits. In state $q1$, the transition that connects $q1$ with $q0$ has the bits needed to be mapped (11). Also, the transition that connects $q1$ with $q2$ can map the same number of bits. In this case, the transition will be chosen randomly, and in this example we choose the one connecting $q1$ with $q0$. Yet after taking such transition, we fail to continue mapping the watermark from state $q0$. This means that we have to return to $q1$ again and continue by choosing the second path, connecting $q1$ with $q2$. Reaching state $q2$, the watermark cannot be mapped to the first bit of the available transition, yet it can be mapped to

[1]A low bandwidth channel that the intruder cannot afford to modify as it uses the most significant components of the object as a means of transition [5].

Authorized licensed use limited to: CONCORDIA UNIVERSITY LIBRARIES. Downloaded on January 7, 2009 at 15:11 from IEEE Xplore. Restrictions apply.

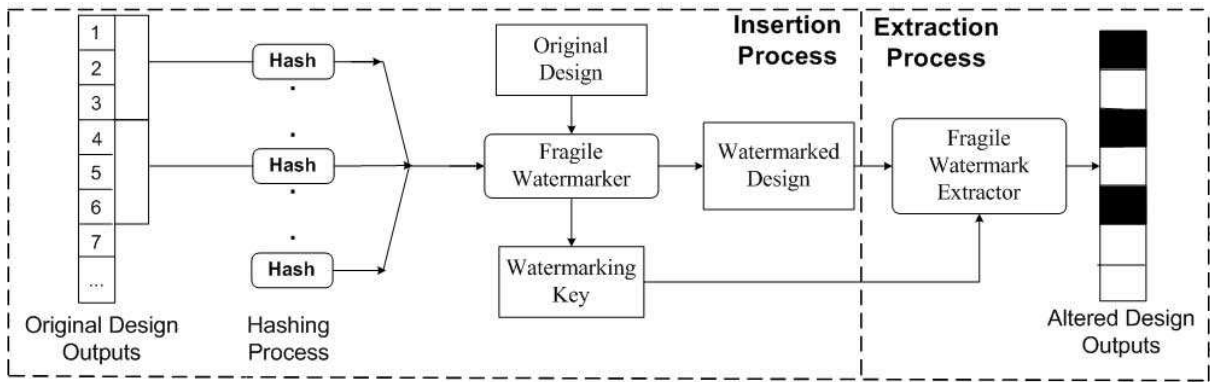
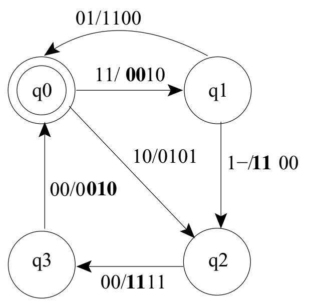


Fig. 1. Fragile Watermarking Framework

the first and second bits (11). Finally, three bits (010) are mapped to the transition connecting $q3$ to $q0$ and finalize the whole watermarking procedure. The algorithm generates the watermarking key shown below the signature which will be used in the extraction phase. The watermarking key consists of a set of triplets. In each, the *first* element represents the input that generates the requested transition, the *second* is the watermark bits that are attached to such transitions, and the *third* element shows the position of the watermark bits in the transition output string.



Signature to be added = **{0 0 1 1 1 1 0 1 0}**

Watermarking Generated Key = (11,00,0),(1−,11,0), (00,11,0), (00,010,1)

Fig. 2. Fragile Watermark: Example

The new algorithm utilizes the same concept of the output mapping technique [2] to embed the hashed watermarking sequence to the design but maps the whole 128 bit signature and not just a part of it. This has to be done because any change of the generated design will affect the previously generated digest. To overcome such a problem, we have embedded a variable number of bits on each transition and embedded these bits in variable positions. This enabled us to map the whole watermark on the STG transitions without adding extra transitions to the system. Starting from a random state of a given STG, the IP watermarking insertion process can be generalized as follows (Figure 3):

1) Compare the first output bit of all transitions of this state to the generated signature to check if they map (bits are equal).
2) In case this bit is equal, we compare the next bit and check which transition has the highest number of coinciding bits to the generated signature.
3) If we reach the maximum number of mapped bits in any transition, the algorithm chooses the transition with the highest number of mapped bits.
4) If two or more transitions happen to have the same number of mapped bits, the algorithm chooses between them randomly.
5) The position of mapped bits as well as their count are recorded to be used as the *watermarking key* in the extraction process.
6) If at any given state the algorithm could not map starting from the first bit, the algorithm tries to map the signature bits starting for the next bit number (second bit of each transition, then the third bit, and so on).
7) If the algorithm fails to find even one mapped bit in any position of a given state, we check the previous state, and change the previously made choice. This is done by using the transition with the second highest number of mapped bits, or by choosing another random transition in case there is more than one transition with equal number of bits.
8) The algorithm will loop until the embedding of all signature bits is finished.
9) If the algorithm fails to embed the whole signature in the end, we choose another random state to start from.

The algorithm uses the least number of transitions possible, by checking all transitions attached to every state and checks which transition can carry the highest number of mapped bits. Yet, this is not sufficient to ensure that we will succeed in embedded the whole sequence. In this case, we choose the second best path, that will lead us to use more transitions but still succeed to embed the watermark. We use iterations if the algorithm failed once, by forcing the algorithm to change the first chosen state.

Authorized licensed use limited to: CONCORDIA UNIVERSITY LIBRARIES. Downloaded on January 7, 2009 at 15:11 from IEEE Xplore. Restrictions apply.

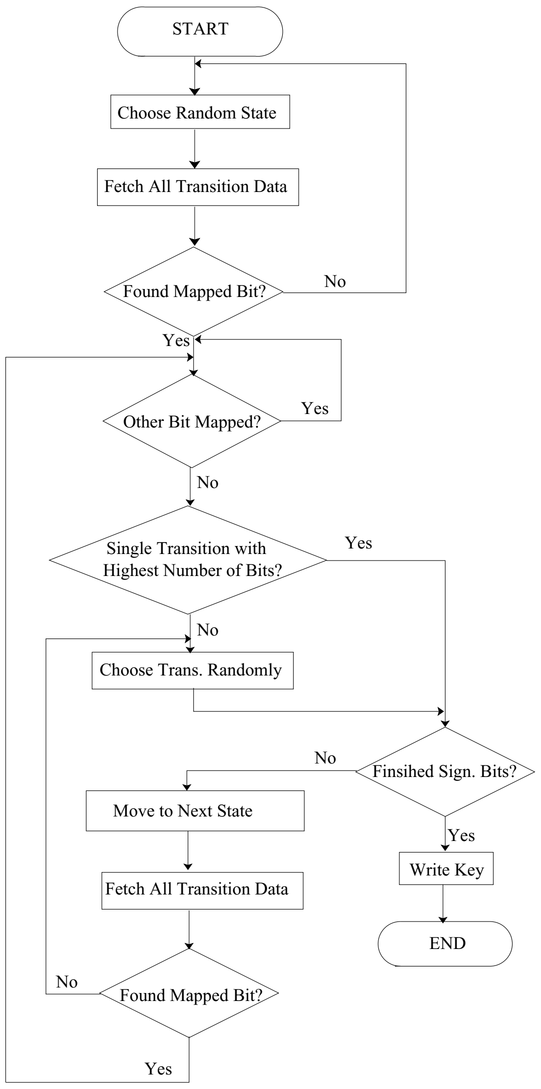


Fig. 3. Fragile IP Watermark Insertion Algorithm

C. Fragile Watermarking Extraction Algorithm

Embedding this signature acts as a carbon paper under the design, because any altering for the design will result directly in a change in the generated digest. In case the customer is worried about the design authenticity the owner can provide him/her with the design digest, and can find out directly if the design is altered, or not using the proposed extraction technique (Figure 4). This can be done in any lower abstraction levels of the design, even after design manufacturing, and the owner does not need to provide the whole higher level design to check authenticity, nor a large complicated testing results.

To extract such digest, we use the extraction algorithm proposed in Figure 4. The watermark extractor uses the perviously generated key, which is composed of the input sequence and the bit positions. This key is used as input to generate the embedded digest (watermark) again. On the other hand, the input file is used to generate the complete output sequence of the system again. This sequence is hashed in the same way we generated the original design digest. Finally, the generated watermark is compared with the extracted watermark from the design to ensure design authenticity. If the generated watermark does not match fully with the new digest, the user can directly pin point parts of the design that were modified or altered.

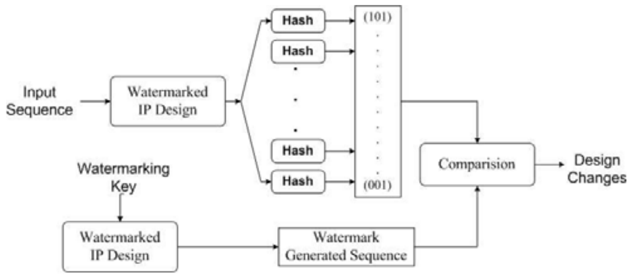


Fig. 4. Fragile IP Watermark Extraction

IV. Watermarking Evaluation and Attack Analysis

The proposed framework is, to the best of our knowledge, the first fragile watermarking technique for hardware designs. Evaluating fragile watermarking approaches is mainly based on the following criteria:

- *Sensitivity for Design Changes* is a measure to how sensitive the watermark is to design changes.
- *Robustness* measures the strength of the hidden mark against attacks and the percentage of undetected watermarked design that might appear.
- *Insertion Overhead* represents the overhead on the design cycle produced by the watermarking insertion technique.
- *Design Overhead* defines the overhead added by the watermarking scheme on the design size.

A watermark insertion should not affect the behavior of the system nor interfere with the original system operation. It is clear that our approach is not inserting any new transitions to the system this means that we are not affecting any extra behavior to the design.

On the other hand, watermark approaches rely on embedding signatures generated from hash functions in order to decrease the watermarking process overhead. In our particular case, the number of bits added by using MD5 hash function is equal to 128 bits. We have mapped every single bit to an already existing bit. This will ensure that there is no design overhead for the fragile watermarking scheme related to area or power. The only overhead introduced to the system by our fragile watermarking scheme is the insertion process overhead that increases the design cycle. Yet, this overhead is extremely small to any other design process. We have measured this overhead using a number of benchmark designs and the results are shown and discussed in the experimental results to follow.

Authorized licensed use limited to: CONCORDIA UNIVERSITY LIBRARIES. Downloaded on January 7, 2009 at 15:11 from IEEE Xplore. Restrictions apply.

Robustness measures the strength of the hidden mark against attacks. Digital watermarking attacks are typically categorized into four main classes [4]: *unauthorized embedding*, *unauthorized removal*, *unauthorized detection*, and *system attacks*.

Embedding attacks or unauthorized embedding aim at embedding another watermark in the design (watermark re-embedding) or try to find a ghost watermark that can be considered an intruder watermark (ghost searching). This attack is meaningless in the case of fragile watermarking, as the intruder is trying to cover his/her design alterations, and not to add new watermarks to the system, especially that this watermark will not help to prove anything in case any design change is found.

Removal attacks aim at the removal of the watermark information. This is attempted without breaking the security of the watermark. Removal attacks are divided into either elimination attacks or masking attacks. In elimination attacks, the intruder tries to completely eliminate the watermark. On the other hand, masking attacks do not aim at removing the watermark itself, but aim at distorting the watermark detector such that it will not be able to detect the availability of the watermark.

Coinciding transitions are considered a supraliminal channels, hence, attacking such transitions or changing any of their values will directly result in destroying the design under investigation. In [2], the authors defined the probability of watermark deletion as "*probability that an attack changes or deletes extra added transitions without deleting at least one original transition*". In our case, we are embedding the watermark by coinciding (mapping) every single transition of it. This means that deleting or changing even one bit of the signature will directly affect the design, or in other words, we are building a perfect supraliminal channel that can never be changed without altering the design, i.e., the removal probability equals 0. This is exactly the main target of the fragile watermarking scheme. It is worth to be mentioned, that although this fragile watermark is effective in detecting design changes, it is complectly inadequate for proving ownership of IP designs. This means that it is requested to use both fragile and robust watermarks complementary to ensure that your IP design is completely secured.

Unauthorized detection of the watermark is considered a part of the removal attacks, as detecting the watermark can be seen as the first step to remove it. Yet, as long as the designer protects his key, the watermark inserted int he IP design cannot be detected by any means.

System attacks aim at attacking the concept of watermarking itself, as an example, attacking the cryptographic base of the watermarking, or removing the chip that checks the watermark physically in case of video for instance. This kind of attacks cannot to be avoided by the watermarking schemes. In our fragile scheme, the main problem is not deleting the watermark, but changing the design in a way that does not affect the embedded watermark, and hence pass undetected. This is the main attack against our technique, and it can be done by generating the exact digest for the watermarked IP, yet such an attack is extremely complicated as the generated signature is based on the strength of the Hash function.

V. Experimental Results

To validate and test the performance of the proposed watermarking algorithms, we have modified the tool developed in [2] in C++ under Windows environment. IP designs are provided as FSMs in the kiss2 format [16], which can be generated by many tools such as SIS [16]. The tool is composed of three main blocks (Figure 5). The *FSM builder* block generates a tree for the FSM representation. The *digest* generator provides the signature to the *watermarker* block after hashing it. Finally, using the *Kiss-to-HDL* block [1], a synthesizeable watermarked VHDL code is generated.

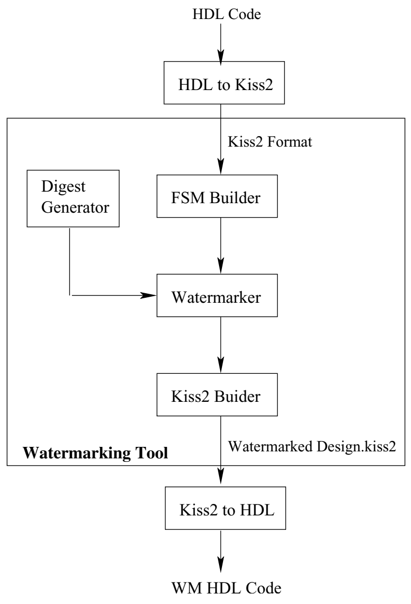


Fig. 5. Watermarking Prototype Tool

To evaluate our approach, the algorithm was applied on the IWLS93 [11] benchmark set using the FSMs generated by the available SIS tool. Tables I describes the results obtained on an Intel Pentium Centrino Duo processor running at 2 GHz and 1 GB of memory. Tables I shows for each design, the number of inputs/outputs, and number of transitions/states. (C/NC) defines if the design under investigation is completely specified (C) or not (NC) taking into consideration that such design should be converted to completely specified ones in real life. (N) represents the number of iterations needed, and (t) is the time (in ms) needed to insert the watermark in each design.

We used the MD5 hash function to generate the fragile signature of each design. We did not divide the design into windows, but we hashed the whole design to generate one single global fragile signature. The table shows that we could insert the fragile watermark in all designs in the test bench

Authorized licensed use limited to: CONCORDIA UNIVERSITY LIBRARIES. Downloaded on January 7, 2009 at 15:11 from IEEE Xplore. Restrictions apply.

with zero overhead, and in a very short time. It worth to be noted that the design *EX1* took the highest insertion time (657 ms) compared to all other designs in the test bench. This is due to the fact that most of the design outputs are equal to zeros, which make it comparatively difficult to insert 128 bits divided between ones and zeros evenly mostly.

VI. Conclusions

Sharing IP blocks in today competitive market poses significant high security risks. Digital watermarking has emerged as a candidate solution at the detection phase of copyright protection for IP blocks. In this paper, we proposed to extend this approach to ensure that designs are not modified by any means. The approach is based on the utilization of coinciding transitions and mapping a digest on such transitions.

We also defined different parameters needed to evaluate the approach and tested it using experimental results with the IWLS93 benchmark. The implemented algorithm is fast and has no overhead on the design, which would ease its integration in the design cycle.

Currently our approach works on flat FSMs, which is a practical drawback for real designs. We are in the course of extending our algorithms to handle hierarchical designs easily. To validate our approach even further, we also intend to apply our tool on industrial sized IP such as PCI bus or Bluetooth protocol.

References

[1] A. Abdel-Hamid, M. Zaki, and S. Tahar, "A Tool for Converting Finite State Machine to VHDL", Proc. of IEEE Canadian Conference on Electrical & Computer Engineering, Niagara Falls, Ontario, Canada, Vol. 4, May 2004, pp. 1907-1910.

[2] A.T. Abdel-Hamid, S. Tahar and E.M. Aboulhamid, "A Public Key Watermarking Techniques for IP Designs"; Proc. Design, Automation and Test in Europe, Munich, Germany, March 2005, pp. 330-335.

[3] 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.

[4] I. J. Cox, M. L. Miller, and J. A. Bloom, "Digital Watermarking", Morgan Kaufmann Publishers, 1998.

[5] S. Cravar, "On Public-key Steganography in the Presence of an Active Warden", Technical Report RC20931. IBM Research Division, T. J. Watson Research Center, Santa Clara, California, USA, July 1997.

[6] S. Fujiwara, G. Bochmann, F. Khendek, M. Amalou, and A. Ghedamsi, "Test Selection Based on Finite State Models", IEEE Transactions on Software Engineering, Vol. 17, Issue 6, June 1991, pp. 591-603.

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

[9] A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe, "Constraint-Based Watermarking Techniques for Design IP Protection", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 20, No. 10, October 2001, pp. 1236-1252.

[10] T. V. Le, and Y. Desmedt, "Cryptanalysis of UCLA Watermarking Schemes for Intellectual Property Protection", In Information Hiding, Lecture Notes in Computer Science, vol. 2578, pp. 213-225, Springer 2002.

[11] K. McElvain, "LGSynth93 Benchmark Set: Version 4.0", http://www.cbl.ncsu.edu/pub/Benchmark_dirs/LGSynth93/, 1993.

[12] N. Narayan, R. D. Newbould, J. D. Carothers, J. J. Rodrguez, and W. Timothy Holman, "IP Protection for VLSI Designs Via Watermarking of Routes", 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 on 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, "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.

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

[16] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. Sangiovanni-Vincentelli, "SIS: A System for Sequential Circuit Synthesis", Technical Report, Dept. of Electrical Engineering and Computer Science, University of California, Berkeley, California, USA, 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] VSI Alliance, "VSI Alliance Architecture Document: Version 1.0", VSI Alliance, 1997.

TABLE I
IWLS93 Benchmark Results

Circuit	**♯I[O]**	**♯T[S]**	**C/NC**	**N**	**t**
MC	3[5]	10[4]	C	1	129
LION	2[1]	11[4]	C	1	78
DK27	1[2]	14[7]	C	3	135
EX4	6[9]	21[14]	NC	1	241
OPUS	5[6]	22[10]	NC	4	243
DK15	3[5]	32[4]	C	1	120
S27	4[1]	34[6]	C	2	98
SSE	7[7]	56[16]	C	2	125
BBARA	4[2]	60[10]	C	1	243
S510	19[7]	77[47]	NC	1	124
S1	8[6]	107[20]	C	1	226
PLANET	7[19]	115[48]	C	1	148
EX1	9[19]	138[20]	NC	20	657
STYR	9[10]	166[30]	C	2	263
S832	18[19]	245[25]	C	4	526
S1494	8[19]	250[48]	C	3	347
S1488	8[19]	251[48]	C	5	630
SCF	27[56]	166[121]	NC	3	965
KIRKMAN	12[6]	370[16]	NC	2	254
S298	3[6]	1096[218]	C	1	139
TBK	6[3]	1569[32]	C	1	182

Authorized licensed use limited to: CONCORDIA UNIVERSITY LIBRARIES. Downloaded on January 7, 2009 at 15:11 from IEEE Xplore. Restrictions apply.