# A Tool for Automatic Watermarking of IP Designs

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

*Abstract*—**Intellectual property (IP) block reuse is essential for facilitating the design process of System-on-a-Chip (SOC). Sharing IP blocks in such a competitive market poses significant high security risks. Digital watermarking, used with most of the shared digital media, has emerged as a candidate solution for helping copyright protection of IP blocks. In this paper, we present an automatic tool for watermarking sequential IP designs. The tool is based on the idea of utilizing unused transitions in the State Transition Graph (STG) to add a part of the watermark. The tool also tries to create a supraliminal channel by utilizing the already existing transitions. The paper describes the structure of the tool, overviews the algorithms used in its components, and reports experimental results obtained by applying it on a set of benchmarks.**

## I. Introduction

Fast advancing integrated circuit (IC) 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. Reusable virtual components or Intellectual property (IP) blocks are most effective when coming to cost reduction and development time of SOC designs. Intellectual property licensing has numerous roots in various media including the printed word, music, art, and machinery. Intellectual property issues would not exist but for the protection of original work from exploitation. Creators and owners of IP designs want assurance that their content will not be illegally redistributed by consumers, and consumers want assurance that the content they buy is legitimate and not forged.

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

The VSI Alliance IP protection development working group [3] identifies three main approaches to secure IPs. First, a *deterrent* approach where the owner uses legal means trying to stop attempts for illegal distribution, i.e., using patents, copyrights and trade secrets. This method does not provide any physical protection to the IP.

Second, a *protection* approach where the owner tries to prevent the unauthorized usage of the IP physically by license agreements and encryption. This approach is used at the distribution phase as well, i.e., the buyer has to have the correct key to decrypt the design and so to use it. Yet, it does not secure leakage from trusted parties, as employees, or brokers.

Third, a *detection* approach where the owner detects and traces both legal and illegal usages of the designs as in watermarking and fingerprinting. This tracking should be clear enough to be considered as evidence in front of a court if needed. The VSI alliance proposed the usage of the three approaches for proper protection of IP designs. The detection approach directly interacts with the IC design, and is considered an overhead on the design cycle. IP watermarking and IP fingerprinting are the main approaches used; where the design is watermarked (tagged) then different tracking techniques are used to keep track of the usages of such design. This is considered to be a passive approach, in which the designer can only track its design, yet it cannot affect its manufacturing except using the watermark evidence in front of the court.

IP watermarking schemes need more development to be integrated in the design cycle. Future IP watermarking schemes should be robust enough to secure the design, but they should not imply a high overhead neither on the design process nor on the final watermarked product. In this paper, we are implementing a previously proposed framework [1] for watermarking sequential IP circuits. The Proposed technique is based on implementing a supraliminal channel [2], by utilizing existing transitions of the covert object (watermarked design). Finite state machines (FSM) are the transformation between inputs and outputs of the design, and can be detected on mostly all lower abstraction levels. FSMs can be represented in many different ways, such as state transition graphs (STG). Making use of such transitions would gives the scheme more strength against different attacks, as well as the ability to detect the watermark in all lower levels.

The rest of the paper is organized as follows: Section II overviews the main approaches used for IP watermarking in the open literature. Section III presents the tool prototype and describes the different blocks of the tool. In Section IV, the watermarking approach used is described as well as the two algorithms used in the tool. Section V shows the experimental results collected by applying both algorithms on a benchmark. Finally, section VI concludes the paper and discusses future work of this on going project.

## II. IP Watermarking: Related Work

There are a few IP watermarking techniques discussed in the open literature: Kahng *el al.* [4] proposed and experimented a *constraint-based* IP watermarking technique. This approach is based on a generic optimizer and the constraint-satisfaction (SAT) problems that can be used on different levels of the design flow. The main advantage of this approach is its low overhead, nevertheless, it has some major drawbacks. Namely, the approach watermarks the solution produced, while it does not watermark the design itself. The watermark cannot be detected except at the same level of abstraction, i.e., tracking the watermark is not easy if the design is resold at other abstraction levels.

At the behavioral level, Oliveira [7], and Torunoglu *et al.* [11] introduced two different techniques used in the watermarking of sequential parts of the design. Both algorithms are based on adding new input/output sequences to the FSM representation of the design. One of the main advantages of both approaches is the ability to detect the presence of the watermark at all lower design levels. These approaches work at a very high abstraction level which provides extra strength. The algorithms are mechanically removable in case of the knowledge of the input sequence and the initial input state. In this paper, we are trying to solve this by presenting a new concept and two new algorithms implementing it. We believe our approach to be the very first public-key watermarking scheme that can be used for watermarking IP designs in a shared industrial environment of intrusted parties.

## III. IP Watermarking Tool

The authors in [1] have proposed an algorithm for watermarking sequential circuits. Their approach relies on the usage of coinciding transitions to increase the watermark robustness as well as decreasing the overhead it might cause on the system. The authors as well have proposed two algorithms that insert the watermark using their novel approach. In this paper, a prototype for both algorithms were built to test their performance. The tool was implemented using C++ under Unix environment. The design accepts *kiss2* [10] standard files (as its FSM representation) which can be generated by many tools such as SIS [10]. Figure 1 shows the structure of the tool which is composed of four main blocks. We start by building a tree for the FSM representation using the ***FSM builder*** block. The ***signature generation*** block provides the signature to the watermarker after hashing it, while the random input and next states needed are provided using a ***random generator*** built in our tool. The three components are added in the ***watermarker***, where the user can choose either of the algorithms to watermark his/her design. Finally, the produced watermarked design is converted again to kiss2 format using a ***Kiss-to-HDL*** block, and the key file that provides the signature added is produced as well.

### A. Watermark Creation: Signature Generator

The proposed model uses a constant length bit sequence as a proof of the ownership rights. The owner should choose any arbitrary length message that will prove his/her ownership and encrypts it using his/her own private key of any encryption algorithm. The encrypted message is then hashed to shorten it to a certain length using a one-way hash function, such as MD5 [9], which produces bits message digest that will be used as an authorship proof (128 bits in case of MD5). This digest is computationally infeasible to find another message to hash the same value or to invert it.

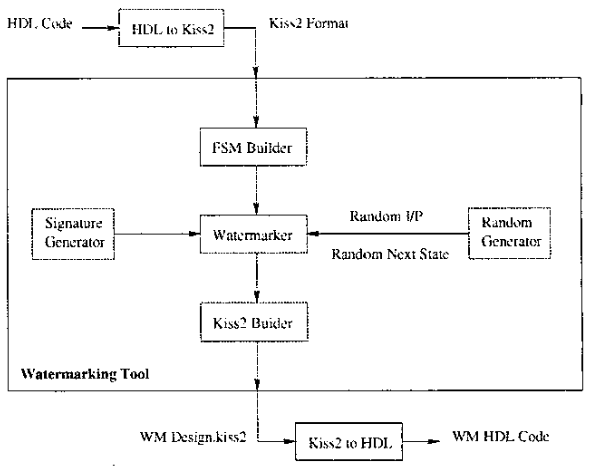

Fig. 1. Watermarking Tool Structure

### B. FSM and Kiss2 Builders

The tool needs to represent the FSM in a tree format to speed the search algorithm. In order to get the FSM in such representation, ***FSM builder*** is integrated in our tool. This block first analyzes the kiss2 file, and extracts different aspects from it, such as number of states, number of transitions, and number of input/output bits. It then builds the tree needed by the ***watermarker*** out of the kiss2 file provided for the design.

After inserting the watermark, ***Kiss2 builder*** rebuild the new kiss2 file. This block takes the watermarked tree and converts it to the primary representation. This block checks the number of inputs, because this number might change due to the watermarking process. Also, extra number of transitions added. This block generates as well the key file that contains the watermark that we should check for.

### C. Input Sequence Generation: Random Generator

Our tool uses a random number generator to produce both the input sequence used and the next transition state in case of added transitions. Usually, randomness is introduced into computers in the form of pseudo-random numbers. For cryptographic use, it is important that the numbers used to generate keys are not just seemingly random; they must be truly unpredictable. Pseudo-random numbers are not truly random. In our case, we relayed on random.org [5], a true random number generating service available on the web, where the atmospheric noise picked up by a radio receiver is considered the source of noise.

### D. Building HDL: Kiss-to-HDL

Kiss2 format [10] is a standard FSM format that is used by many tools. Kiss2 contains a basic and clear description for the FSM. The final block of the tool converts the kiss2 representation of the FSM design to VHDL [12] sequential code. This can be done by converting all the states and transitions of the FSM to a conditional VHDL program used directly in the hardware design, or as a part of a larger design. This block allows easier integration of our tool into the design cycle.

## IV. Watermarking Insertion Techniques

The implemented ***watermarker*** is based on the Coinciding Transitions approach proposed in [1]. The approach defines the ownership rights as an output sequence of bits, that is divided into input/output pairs. These pairs are added to the design to build the ownership proof by either, adding extra transitions in case the transition is not used, or trying to coincide this pair with an already existing one to increase robustness. Thus, the signature is built using the free transitions available in the system, i.e., utilizing the difference between complete FSMs and non-complete FSMs proposed in [11]. Also, in order to increase robustness of the watermark, a supraliminal channel is set on the already existing transitions, i.e. the signature is inserted by utilizing existing transitions that will coincide with the signature. Making use of such transitions gives the system extra strength against attacks [1], and decreases the watermark overhead to the system. Finally, the approach minimizes the search needed for inserting the watermark, which is directly related to the time needed for watermark insertion. The next subsections describes the two different algorithms that were implemented in the ***watermarker***.

### *A. Input Comparison Algorithm*

The output generated sequence will be associated with a random generated input and will be consider a new pairs in the transition set. Starting form an arbitrary state, these sequences will be added to the STG according to the following algorithm:

1) The random input is compared with the inputs of the selected state to check if such input is already defined in the STG. nets 2) In case this input is not used, as in our case, an extra transition is added directly to the STG with a randomly chosen next state.

3) In case this input is already used the output sequence is checked to see if the output coincides with the generated signature. The transition will be considered as part of the signature, and the algorithm will advance to the next state.

4) Finally, if the output does not match the one there, and there are no free inputs available, the algorithm adds an extra input bit to the STG. The logic value '0' is assigned to the already existing transition, and the logic value '1' will be used for the watermark transition need to be added.

The algorithm does not search the system states of the STG to insert the watermark, this would not cause high overhead on the design flow. YET, the algorithm is not trying to maximize the coinciding transitions, but it is a kind of best-effort algorithm that randomly finds coinciding transitions.

### *B. Output Mapping Algorithm*

In this approach, we utilize more search efforts in the algorithm. We try to increase the number of the coinciding transitions in order to increase the robustness. This was done by forcing the algorithm to search the output bits of the STG before the input ones and take the watermarking decision depending on this search.

The output signature is divided in sub-sequences with the same number of output bits, then starting for an arbitrary state, the signature is added according to this algorithm:

1) The signature output bits will be compared to all the outputs of the state to see if any of them would coincide.

2) In case of such condition being satisfied, the transition will be considered as a part of our STG, and the next state will be determined through such transition.

3) If such an output does not exist, the tool will search the inputs domain of such state to find any free input sequence and add an extra transition using such free input.

4) Finally, if all the inputs are already being used, an extra input bit will be added to extend the whole STG converting it to a non complete FSM and gives more room for the transition needs to be added. This bit will be forced to (0) in the case of already existing transitions, and (1) for added transitions, such extra transition.

Using this algorithm, the coinciding transitions will be maximized for any given state. Also, the number of added input bits should be decreased. This approach will introduce more time overhead to the design phase as the outputs of the FSM are searched for every visited state.

## V. Experimental Results

We applied both algorithms in our prototype on the IWLS93 benchmark set [6] using the FSMs generated by the available SIS tool to convert the code to kiss2. Table I describes the results obtained by applying the Input Comparison Algorithm on the benchmark set using a Sun Sparc Ultra 5 machine. The total number of transitions of each design, and the total number of added transitions $m$ is shown. $m$ is divided into the number of extra added transitions ($m_1$) and the number of coinciding transitions ($m_2$). $e_{wm}$ is the number of extra inputs needed to add the watermark. Finally, the table shows the time used to insert the watermark in each design. It is clear that the watermark can be considered as a real overhead in the small designs (as in s27 where the number of total transitions is 34 compared to 128 added transitions). Yet, as the designs get larger, the overhead decreases significantly, as in the case of *tbk*, where 43 extra transitions are added compared to 1569 existing transitions.

Table II describes the results obtained by applying the Output Mapping Algorithm using the same machine. The number of coinciding transitions ($m_2$) increased significantly in many cases (from 6 to 18 in the case of *tbk* for example). Also, the time increases mostly to add the watermark in this case.

## VI. Conclusions

Sharing IP blocks in today's competitive market poses significant high security risks. Different approaches were defined trying to decrease such risks in a fast growing market. Digital watermarking has emerged as a candidate solution for helping copyright protection of IP blocks.

In this paper, we have implemented a sequential IP watermarking tool based on the coinciding transitions in FSMs. The approach utilizes coinciding transitions as well as the unused transitions in order to give high robustness to the watermarked design. We implemented two main algorithms to embed the signature in the design. First, Input Comparison

TABLE I
IWLS93 BENCHMARK RESULTS USING THE INPUT COMPARISON ALGORITHM

| Circuits | ♯ states | ♯ I/O | ♯ trans. | $e_{wm}$ | $m$ | $m_1$ | $m_2$ | t(ms) |
|---|---|---|---|---|---|---|---|---|
| s27 | 6 | 4/1 | 34 | 1 | 128 | 19 | 109 | 34ms |
| bbara | 10 | 4/2 | 60 | 2 | 65 | 20 | 45 | 15ms |
| dk14 | 7 | 3/5 | 56 | 3 | 26 | 25 | 1 | 9ms |
| styr | 30 | 9/10 | 166 | 2 | 13 | 12 | 1 | 15ms |
| bbsse | 16 | 7/7 | 56 | 2 | 19 | 17 | 2 | 9ms |
| cse | 16 | 7/7 | 91 | 2 | 19 | 17 | 2 | 10ms |
| sse | 16 | 7/7 | 56 | 2 | 19 | 17 | 2 | 10ms |
| ex1 | 20 | 9/19 | 138 | 2 | 7 | 7 | 0 | 10ms |
| scf | 121 | 27/56 | 166 | 1 | 3 | 3 | 0 | 54ms |
| s420 | 18 | 19/2 | 137 | 2 | 65 | 43 | 22 | 33ms |
| tbk | 32 | 6/3 | 1569 | 2 | 43 | 37 | 6 | 51ms |

TABLE II
IWLS93 BENCHMARK RESULTS USING THE OUTPUT MAPPING ALGORITHM

| Circuits | ♯ states | ♯ I/O | ♯ trans. | $e_{wm}$ | $m$ | $m_1$ | $m_2$ | t(ms) |
|---|---|---|---|---|---|---|---|---|
| s27 | 6 | 4/1 | 34 | 1 | 128 | 1 | 127 | 9ms |
| bbara | 10 | 4/2 | 60 | 2 | 65 | 20 | 45 | 15ms |
| dk14 | 7 | 3/5 | 56 | 3 | 26 | 23 | 3 | 15ms |
| styr | 30 | 9/10 | 166 | 2 | 13 | 12 | 1 | 21ms |
| bbsse | 16 | 7/7 | 56 | 2 | 19 | 16 | 3 | 15ms |
| cse | 16 | 7/7 | 91 | 2 | 19 | 17 | 2 | 16ms |
| sse | 16 | 7/7 | 56 | 2 | 19 | 16 | 3 | 14ms |
| ex1 | 20 | 9/19 | 138 | 2 | 7 | 7 | 0 | 11ms |
| scf | 121 | 27/56 | 166 | 1 | 3 | 3 | 0 | 68ms |
| s420 | 18 | 19/2 | 137 | 2 | 65 | 8 | 57 | 78ms |
| tbk | 32 | 6/3 | 1569 | 2 | 43 | 25 | 18 | 81ms |

algorithm, which mainly search inputs of each state. It is a best effort approach that uses the coinciding transition if it finds it. Second, Output Mapping Algorithm, searches each state it visits trying to find a coinciding transition to embed the watermark in.

We tested our tool on a set of benchmark circuits. The tool tends to work fine on this size of circuits, yet it needs to be tested and applied on larger designs. Beside, our tool mainly works on flat FSMs, which is mostly not the case when it comes to complicated designs needed to be watermarked. Finally, although, the second implemented algorithm searches the outputs for any available coinciding states, we think that the algorithm can be more efficient by having a kind of speculation so that, the next state would be the one that might have higher probability of coinciding transitions.

## REFERENCES

[1] A. T. Abdel-Hamid, S. Tahar and E. M. Aboulhamid, "A Frame Work for Watermarking IP Sequantial Designs", Technical Report, Electrical and Computer Engineering Department, Concordia University, Montreal, Quebec, Canada, February 2004.

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

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

[4] 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 International Conference on Computer-Aided Design*, San Jose, California, USA, November 1999, pp. 600-604.

[5] M. Haahr, "Introduction to Randomness and Random Numbers", *www.random.org*, June 1999.

[6] K. McElvain, "LGSynth93 Benchmark Set: Version 4.0", $www.cbl.ncsu.edu/pub/Benchmark_airs/LGSynth93/$, 1993.

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

[8] 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.

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

[10] 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,CA 94720, 1992.

[11] 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.

[12] IEEE standard 1076-1993, "IEEE Standard Description Language Based on the VHDL Hardware Description Language",1993.