A Public-Key Watermarking Technique for IP Designs

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

***Abstract*— Sharing IP blocks in today's competitive market poses significant high security risks. Creators and owners of IP designs want assurances that their content will not be illegally redistributed by consumers, and consumers want assurances that the content they buy is legitimate. Recently, digital watermarking emerged as a candidate solution for copyright protection of IP blocks. In this paper, we propose a new approach for watermarking IP designs based on the embedding of the ownership proof as part of the IP design's FSM. The approach utilizes coinciding as well as, un-used transitions in the state transition graph of the design. Our approach increases the robustness of the watermark and allows a secure implementation, hence enabling the development of the first public-key IP watermarking scheme at the FSM level. We also defin evaluation criteria for our approach, and us experimental measures to prove its robustness.**

I. Introduction

Fast advancing IC (integrated circuit) processing technologies have enabled the integration of full systems an a single chip forming the new paradigm of the "System-on-a-Chip" (SOC) technology. Reusable virtual components or Intellectual Property blocks (IPs) are most effective when it comes to reducing cost and development time of SOC designs. IP designs are a big investment for companies, on the other hand, sharing IP designs pose significant high security risks. Most of these IPs need time and effort to be designed and verified, but they can be copied, or modified to cover the authorship proof. Creators and owners of IP designs want assurances that their content will not be illegally redistributed by consumers, and consumers want assurance that the content they buy is legitimate.

The VSI Alliance IP protection development working group [7] identifies three main approaches to secure IPs: deterrent, protection, and detection. The first uses legal means, such as patents, copyrights or trade secrets to stop attempts for illegal distribution. The second prevents the unauthorized usage of the IP physically by license agreements and encryption. The third detects and traces both legal and illegal usages by using watermarking and fingerprinting. Protection techniques do not track the design in order so check if the buyer resold it, or even reused it without permission. They also cannot stop local leakage, such as employees or sub-contractors. Like digital media, watermarking IPs emerged as the most effective solution for tracking IP designs after sales, where the owner has a strong enough evidence to be used in front of court.

In this paper, we propose an IP watermarking approach that can be used early in the design cycle, and which is based on the finite state machine (FSM) of the IP. FSMs model the transformation between inputs and outputs of the IP design and can be represented in different forms, such as state transition graphs (STG). FSM watermarking enables the detection of inserted watermark at lower implementation levels.

We propose a watermarking approach in which both existing and unused transitions in the FSM are used to embed the signature. In case no free transitions are available (completely specified FSM, CSFSM), we add extra input bits to insert the watermark. Using existing transitions provides a supraliminal channel [1] as it would give more strength to the system against different attacks. It also helps balancing between adding enough data to identify the owner and the design overhead (area, power, delay) this data may introduce.

The rest of the paper is organized as follows: Section II gives a brief description of related work on IP watermarking. Section III presents our IP watermarking and extraction techniques. Section IV provides measures used to evaluate the performance of our approach, and to analyze the attacks. Section V presents a prototype implementation of the algorithm, including different experimental results. Finally, Section VI concludes the paper.

II. Related Work

There are many IP watermarking or fingerprinting techniques available in the open literature. They fall into two classes [1]. *Dynamic* watermarking, where the watermark is detected by running the watermarked IP and detect the generated signal. *Static* watermarking, where the watermark is considered as a property of the design, and can only be detected by different static techniques.

Kahng *el al.* [8] 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 the problem. The approach was

[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]

1530-1591/05 $20.00 © 2005 IEEE

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 *et al.* [6]. At lower design levels, the approach was used even more effectively in routing [11], placement, and floor planning [8].

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.* [9] showed that several constraint-based watermarking schemes can be broken easier than previously thought.

At the behavioral level, Oliveira [12], and Torunoglu *et al.* [16] 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. The main advantage of both approaches is the ability to detect the presence of the watermark at lower design levels.

The algorithm in [16] 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 [12] 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. The extra states added can be removed using a state reduction approaches. The author in [12] proposed to solve this problem by slightly changing the functionality of the STG. This also cannot be done mechanically as it might affect the design functionality.

III. Watermarking FSMs Using Coinciding Transitions

As a passive technique, one of the main challenges of watermarking schemes is the authenticity of the watermark. In the scheme we propose in this paper, this problem is solved by using a secure third party, e.g., a watermarking governing body. This governing body will be responsible for generating and distributing time-stamped authenticated signatures, as well as keeping a record for such signatures for the extraction phase.

The proposed watermarking scheme is composed of three main parts: *Signature generation, watermark insertion (embedding),* and *watermark detection (extraction)*. The watermark embedding phase is done by the designer, where he/she uses the authentic signature to embed the watermark using the embedding algorithm proposed below. Finally, in the manufacturing facilities and afterwards, the designer introduces the key needed to detect the watermark, in order to prove the authenticity of the design.

A. Signature Generation

The signature generation is done by the watermarking authority (third party which will prevent intruders from searching for ghost watermark and consider it as their watermark (known as ghost attacks). The generated signature will be also time-stamped to prevent intruders from re-embedding their watermark in the system.

The secure third part will use the ownership information provided by the IP designer and encrypts it using any public/private-key encryption algorithm after time-stamping it. The encrypted information is then hashed giving a short digest to decrease the watermark embedding overhead. This digest is computationally infeasible to find another message hashes the same value.

In our proposed model, the owner chooses any arbitrary length message that will prove his/her ownership and encrypts it using his 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, MD5 [14] in our case, to generate a constant length bit sequence (128 bits) as a proof of ownership, if using MD5.

B. Coinciding Transitions Approach: Output Mapping

The watermark insertion algorithm attempts to coincide a part of the watermark on the STG transitions to increase the watermark robustness. This is done by searching different outputs of 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. Starting form any randomly chosen state (S_x), the watermark will be added to the STG according to the following steps:

1) Compare the outputs of the state S_x to the generated signature to check if they coincide.
2) In case one of the outputs is equal to the watermark bits, this transition will be considered part of our watermark.
3) If the signature sequence is not equal to any of the outputs, then the inputs of S_x will be checked to determine if there is any free input that can be used to add an extra transition. The next state in this case will be chosen randomly, with preference given to states with free transitions.
4) In case all the inputs are already being used, an extra input bit e^i_{wm} is added to the system to extend the FSM. This input bit will have the same logic value for all already existing transitions. For instance, a logic value '0' assigned to the already existing transitions and logic value '1' will be used for the watermark transition added. The next state will be chosen randomly.
5) The algorithm will loop until embedding all the signature bits is done.

Figure 1 illustrates an example for the above algorithm using the signature in the bottom of the figure. Starting from state ($S0$), the tool finds a coinciding output (00) and moves to S3. In S3 no putput 11 exists but input 00 is free. The next state in this case will be decided randomly and the algorithm advances to state (S2). In S2, all inputs are being used, hence an extra input bit is added to extend the whole STG. This bit will be forced to be equal to "0" for existing transitions out of S2 and "1" for added ones. This extra transition will drive the STG to state (S0) randomly as well.

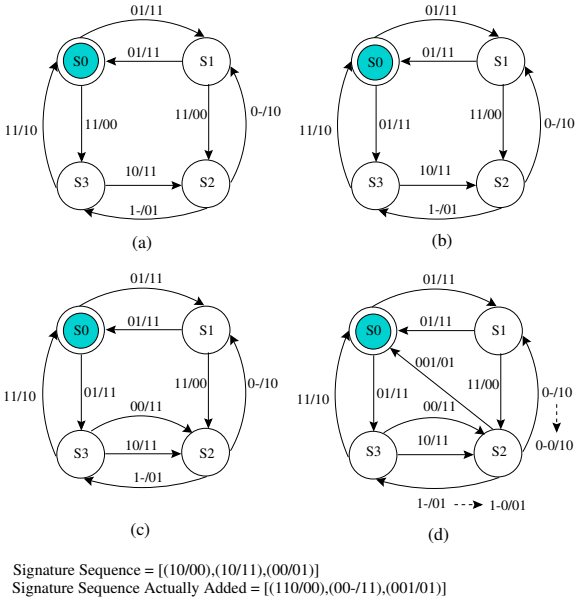

Fig. 1. Output Search Watermarking Algorithm: Example

The algorithm does not search the system states of the STG to insert the watermark, it simply inserts the watermark directly on a randomly chosen state. This makes it fast and would not cause high overhead on the design flow. It was noted that the number of coinciding transitions will decrease as the number of outputs increases. This happen as the probability to find a number of outputs equals coincide with the watermark decreases as the number of output increases. To solve such problem, the algorithm works into iterations. Each iteration works with different number of outputs, and tries to coincide more transitions. Afterwards, the tool decides between the generated solutions based on the robustness, as well as lower design overhead introduced, as discussed in the next section.

C. Watermark Extraction

The third phase of the watermarking process is tracking the watermarked design. In the proposed watermarking technique, direct detection can be used through the previously generated input in the watermarking process to generate the output signature. Using direct detection, as proposed in [16], will not enable the owner to detect the output in the correct form [1] in case of masking attacks, discussed in details in the next section. Traces of the watermark still exist and it is enough to be considered as an evidence in front of a court. In such case, rebuilding the whole FSM is the ultimate solution to extract all the traces left from the watermark, but this is an expensive and complicated task.

In our approach, coinciding transitions cannot be deleted. This will give the system extra robustness as the attacker will have to decide between used and unused transitions. To solve masking attacks, we propose an extraction algorithm making use of coinciding transitions as marks, or semaphores, to detect if the watermark traces exist. The extraction algorithm is as following:

1) During the embedding of the watermark, the tool builds an extra file containing different input/outputs pairs (different sequence paths) that lead to the coinciding transitions.
2) During the extraction, if direct detection fails. The extraction tool will start by checking the previously defined paths and check the availability of coinciding transitions.
3) To find a coinciding transition, the tool will search for all the extra added transitions that might exist around this state.
4) The extraction tool searches all the coinciding transitions and extracts all non-deleted extra transitions in the system.

This algorithm will force the attacker to delete all extra added transitions in the system. A very hard process with a very low probability, defined as a measure of robustness in the next section. The algorithm still needs to be optimized in order to decrease the search time. Yet, it shows an extra advantage of using coinciding transitions.

IV. Evaluating Watermark Performance

Petitcolas [13] identified a set of measures for watermark evaluation. These measures were developed mainly for multimedia applications but still some of them are essential for the evaluation of IP watermarking techniques. These measures are described briefly in the next subsections. For further information about attack analysis and other measures the reader can refer to [1].

A. Impact on Design Functionality

Perceptibility [13] is a measure of how much the hidden mark has deteriorated the perceived quality of the system. In hardware design, designers cannot accept behavior changes in a system due to watermark insertion, i.e., watermarking technique should not interfere with the original system operation. The following theorem proves the soundness of the watermarked design automaton M_{wm} with respect to the original behavior M.

Theorem 1*: The watermarked design M_{wm} behaves exactly as the original design M for any set of arbitrary inputs, under the condition that all extra added bits e_{wm}, used for watermarking, are set to the same secure logical level defined at the watermarking insertion stage.*

Let $\tilde{a}$ be any arbitrary input sequence composed of *m* elements, such that $\tilde{a} = (a^0, a^1, ...a^{m-1})$. $\tilde{a}_{wm}$ is the same input sequence for the watermarked design, where $\tilde{a}_{wm} = \langle \tilde{a}, e_{wm} \rangle$, then:

$$\forall\ \tilde{a}\ e_{wm}.\ \ e_{wm} = C \rightarrow\ (\lambda(q^0, \tilde{a})\ =\ \lambda_{wm}(q^0_{wm}, \tilde{a}_{wm}))$$

where C is a secure constant logical value for all extra added bits pre-defined at the time of inserting the watermark, q^0_{wm} is the initial state of the watermarked design. The theorem is proved mathematically in [1]. It is proved that under the above conditions, the initial state in both designs will behave identically. And then proved that the output functions (λ and λ_{wm}) will produce the same values for any given set of input sequence.

B. Reliability and Attack Analysis

The level of reliability [13] is divided into two main aspects. *Robustness*, which measures the strength of the hidden mark against attacks, and *false positive*, which defines the probability a watermark detector can find an ownership mark in a non-watermarked design. In the case of public-key operation, the level of reliability should include more measures for *asymmetry robustness* [13].

1) Attack Analysis: Digital watermarking attacks are categorized in four main classes [4]: *unauthorized removal*, *unauthorized embedding*, *unauthorized detection*, and *system attacks*. The same categorization applies for IP watermarking schemes. *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. Yet, the VISA IP protection group solves this by protecting the design through different transactions.

Embedding attacks, 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). As for most of the well known watermarking schemes, we are proposing a third entity (a secure third party) that is responsible for granting and authorizing an embedded time-stamped signature (ownership certificate). These two measures will prevent both embedding attacks, because the intruder will be forced to extract a 128 pre-defined bits (if using MD5) from the system, that totaly coincide with his signature. The probability of finding ghost signatures is directly related to the probability of finding false positives as it will be discussed in the next subsection, and is extremely small. On the other hand time-stamping will stop the intruder from re-embedding his/her watermark in a previously licensed system, because the database of the third entity will directly reveal the real owner.

Removal attacks, [4] 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. Coinciding transitions are considered as supraliminal channels. The probability of watermark deletion is defined as following "*probability that an attack changes or deletes extra added transitions without deleting at least one original transition*".

This probability will differ depending on both detection and operation modes. In the symmetric (secret) mode, this probability will be related to the total number of design transitions (n), in the case of asymmetric mode, the probability will be related to the number of the watermark transitions, as the intruder will be able to know these transitions.

To Remove the watermark, the intruder need to delete most of the extra transitions (m_2) without deleting any of the originally existing transitions (n) to delete the watermark. Thus, the removal probability of the watermark (P_r) defined as " *probability that any attack would change or delete all extra added transitions (m_2) without deleting at least one original transition*". As discussed above this probability depends on the mode of operation (symmetric or asymmetric). In the symmetric mode, the intruder needs to choose from n existing transitions, so that P_r^s is defined as:

$$P_r^s \;=\; \frac{1}{C_{n+m_2}^{m_2}}$$

where, C_j^k is the combination of k and j, such that $C_j^k = j!/(k! \times (j-k)!)$.

On the other hand, in the public-key organization, the intruder does not need to choose from the n originally available transitions, but from m watermarking transitions. This means that the removal probability (P_r^a) here will be calculated as:

$$P_r^a \;=\; \frac{1}{C_m^{m_2}}$$

Asymmetric techniques in general are not considered as robust as symmetric ones [13]. The main measure of our approach capability to work in an asymmetric (public-key) mode is P_r^a. Depending on such measure, the system cannot work in the public mode unless this probability is smaller than a certain value defined by the designer. This is done, by choosing between different iterations that will satisfy the probability condition and has the minimal design overhead. It is worth to be mentioned that a second secret watermark can be added to the system in case the intruder could break the public one. This will add extra overhead to the system, but will be rewarded by a higher level of security.

2) Detecting False Positives: The probability of coincidence, is defined as a main measure of the authenticity of the watermark. This probability is considered as a measure for detecting the watermark in a design by accident in a non-watermark design (false positives). In [16], the probability of coincidence (P_u) for an FSM was defined as "*the odds that an unintended watermark is detected in a design*". It is also considered a measure for the ghost attacks discussed above. This probability was calculated under the approximation that all the transitions have the same probability of occurrence as:

$$P_u = \frac{1}{[2^{|\Delta|}]^x - 1} \quad n \geq 1$$

where x is the number of extra added transitions and $\mid \Delta \mid$ is the total number of possible outputs. In our system, we are only adding m_2 extra transitions, but the owner still needs a sequence of length m to detect the watermark.

Using the MD5 hash function, a constant number of bits for the watermarking sequence (128 bits) is introduced. m is calculated as the upper limit of the division of the number of added bits by the number of outputs, i.e., $m = \lceil \frac{128}{|\Delta|} \rceil$. We can then calculate the lower bound of the coinciding probability, the worst achievable case, ($\bar{P_u}$) as:

$$\bar{P_u} \;=\; \frac{1}{[2^{|\Delta|}]^{\lceil \frac{128}{|\Delta|} \rceil} - 1}$$

Hence,

$$P_u \;\simeq\; \bar{P_u} \;=\; \frac{1}{2^{128}-1} \;=\; 2.938 \times 10^{-39}$$

This means that we can safely state, that our probability of coincidence is nearly constant and is larger than the above

value for the 128 bits signature used. If the designer needs a lower value for P_u, he/she can either change the hash function used, or re-watermark the design using the same technique again with a second signature.

C. Watermarking Overhead

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. This amount can be increased at the expense of the number of input bits as well as the area and extra logic added to the system. Mapping more coinciding transitions directly means that we will have less overhead. The watermarking overhead is divided into three different issues: *area*, *delay*, and *power*. We have measured the percentage of increase on a number of benchmark designs before and after watermarking. The results are shown and discussed in the experimental results to follow.

V. Implementation and Experimental Results

A. IP Watermarking Tool

A Prototype for the algorithm was built to test its performance. The tool [3] was implemented using C++ under Unix environment. The design accepts FSMs in kiss2 format [15], which can be generated by many tools such as SIS [15]. Figure 2 shows the different implementation blocks. The tool is composed of four main blocks. We started by building a tree for the FSM representation using the *FSM builder* block. The *signature generation* block provides the signature to the watermarker block after hashing it, while the random input and next states needed are provided using a *random generator* built in our tool. In the *watermarker* block, the user can choose the number of iteration the algorithms can run to watermark his/her design. The watermarker block also includes a decision block that chooses between iteration results using the highest probability P_r^a that is smaller than a certain value defined by the user (10^{-6} in our case). The choice takes into consideration the lowest overhead possible (least added transitions, and lowest number of added bits). Finally, using the *Kiss-to-HDL* block [2], a synthesizeable watermarked VHDL code is generated.

B. Experimental Results and Comparison

To evaluate our approach, the algorithm was applied on the IWLS93 [10] benchmark set using the FSMs generated by the available SIS tool. Table I describes the result obtained on a Sun Sparc Ultra 5 machine with a 256 MHz Processor and 512 MB of memory. All the circuits were synthesized using Synopsys Design Analyzer on the same machine. It shows for each design, the number of states, inputs/outputs, transitions. The total number of added transitions (m), coinciding transition (m_1), extra added transitions (m_2), and extra inputs needed to add the watermark (e_{wm}) are also shown. C/NC defines if the design under investigation is completely specified, so at least 1 input is needed, or not. N represents the number of iterations. Finally, the time (t) needed to insert

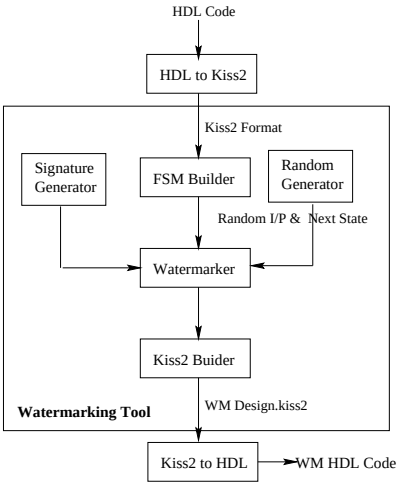

Fig. 2. Watermarking Prototype Structure

the watermark in each design is given in *ms*. The algorithm adds the transitions without building the whole design tree, to decrease the time needed to watermark a design. On the other hand, the random generator favors states with free transitions, higher seeds for such states, but this did not prevent uin adding an input although the design is NCFSM, such as in the case of EX1. The time for inserting the watermark is extremely short, hardly exceeded 8 seconds in the case of *scf* with 54 iterations involved, which gives a good indication about the low overhead the algorithm can introduce in the design cycle.

Table I evaluates the performance of the algorithm. This is done by calculating the different probabilities shown before in section IV. The removal probabilities (P_r^s and P_r^a) are shown. Table I shows the area, power, and delay overhead compared to the synthesized original circuit. The experimental results demonstrate that the table that our approach has a very low effect on the delay and power, especially when the design tend to get larger. On the other hand, the area has high overhead for small designs, then decreases as the designs get larger. This is due to the large signature size compared to the original circuit. As for the robustness of the system, the algorithm failed to watermark some designs efficiently, either it could not coincide transitions, such as for S1488, which cannot operate in public-mode. This can be solved by changing the signature generated or re-watermark the design, but this will result on higher design overhead.

It is worth mentioning that, the probability of coincidence P_u is nearly constant and less than 2×10^{-39}. Also, increasing the coinciding transitions (m_2) will directly increase the robustness of the system in the public-key case, yet it lowers the robustness of the system in the secret-key mode. Obviously, the overheads decreases, as the design size increases.

VI. Conclusions

Sharing IP blocks in such a 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, analyzed,

and implemented a novel approach for watermarking sequential IP designs. The approach was based on the utilization of coinciding transitions as well as the unused transitions of the design FSM in order to give higher robustness. The approach works in a public-key organization, where the detection key can be shared with non-trusted parties.

We also defined different parameters needed to evaluate the approach and tested it using experimental results with the IWLS93 benchmark. The implemented algorithms are fast and have a comparatively low overhead on the design, which would help it to be integrated easily in the design cycle.

The main drawback of our approach is that it works on flat FSMs, which is not the case when it comes to real designs. We are in the course of extending the approach to handle hierarchal designs, in order to ease its integration.

REFERENCES

[1] A. T. Abdel-Hamid, S. Tahar and E. M. Aboulhamid, "Hardware IP Watermarking for Copyright Protection", *Technical Report, Electrical and Computer Engineering Department*, Concordia University, Montreal, Canada, June 2004.

[2] A. T. 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 (CCECE'04), Niagara Falls, Ontario, Canada, Vol. 4, May 2004, pp. 1907-1910.

[3] A. T. Abdel-Hamid, S. Tahar and E.M. Aboulhamid: A Tool for Automatic Watermarking of IP Designs; Proc. IEEE Northeast Workshop on Circuits and Systems (NEWCAS'04), Montreal, Quebec, Canada, June 2004, pp. 381-384.

[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, July 1997.

[6] I. Hong and M. Potkonjak, "Techniques for intellectual property protection of DSP designs", Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing, Munich, Germany, April 1997, pp. 3133-3136.

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

[8] 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", Proc. of IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 20, No. 10, October 2001, pp. 1236-1252.

[9] T. V. Le, and Y. Desmedt, "Cryptanalysis of UCLA Watermarking Schemes for Intellectual Property Protection", Lecture Notes In Computer Science, 5th International Workshop on Information Hiding, Noordwijkerhout, The Netherlands, October 2002, pp. 213 - 225.

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

[11] 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.

[12] 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.

[13] F. A. P. Petitcolas,"Watermarking Schemes Evaluation", IEEE Magazine of Signal Processing, Vol. 17, No. 5, September 2000, pp. 58-64.

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

[15] 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 94720, Dept. of Electrical Engineering and Computer Science, University of California, Berkeley, USA, 1992.

[16] 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.

TABLE I

IWLS93 BENCHMARK RESULTS USING OUTPUT MAPPING ALGORITHM

Circuit	**♯I[O]**	**♯T[S]**	e_{wm}	***m***	m_1	m_2	**C/NC**	**N**	**t**	P_r^s	P_r^a	$Area\%$	$Power\%$	$Delay\%$
MC	3[5]	10[4]	2	33	17	16	C	3	455	3.770e-12	8.570e-10	323	28.6	2.4
LION	2[1]	11[4]	1	128	116	12	C	1	35	1.500e-17	4.214e-17	240	17.307	0
DK27	1[2]	14[7]	2	64	48	16	C	1	85	5.814e-17	2.046e-15	153.2	19	3.6
EX4	6[9]	21[14]	0	26	13	13	NC	6	416	E7.1084e-12	9.614e-8	29.3	23.4	5.8
OPUS	5[6]	22[10]	0	22	9	13	NC	4	149	1.926e-11	2.010e-6	34.9	13.2	6.3
DK15	3[5]	32[4]	1	34	21	13	C	3	146	4.890e-14	1.077e-9	92.8	17.2	1.8
S27	4[1]	34[6]	1	128	125	3	C	1	133	1.4377e-6	2.929e-6	36.2	3.7	0
SSE	7[7]	56[16]	1	19	6	13	C	5	214	1.268e-24	5.918e-15	29.2	14.2	2
BBARA	4[2]	60[10]	1	64	42	22	C	1	27	7.1750e-25	1.244e-17	72.3	17.3	-1.1
S510	19[7]	77[47]	1	22	8	14	NC	4	286	2.631e-17	3.127e-6	34.2	8.9	3.2
S1	8[6]	107[20]	1	28	13	15	C	4	485	3.251e-20	2.671e-8	31.7	14.8	3.4
PLANET	7[19]	115[48]	1	9	4	5	C	17	829	4.441e-9	0.0079	36.2	12.4	0.9
EX1	9[19]	138[20]	1	8	3	5	NC	17	1001	1.938e-9	0.0178	17.391	14.516	3.5
STYR	9[10]	166[30]	1	15	7	8	C	8	566	4.094e-14	1.554e-4	13.4	15.4	1.8
S832	18[19]	245[25]	1	7	0	7	C	17	743	1.236e-13	1	6.8	6.4	0
S1494	8[19]	250[48]	1	8	7	1	C	17	646	8.494e-14	0.125	11.6	3.2	1.8
S1488	8[19]	251[48]	1	7	0	7	C	17	1407	0.0038	1	14.6	3.2	0
SCF	27[56]	166[121]	0	3	0	3	NC	51	8021	1.265e-6	1	3.5	9.8	1.4
KIRKMAN	12[6]	370[16]	0	22	11	11	NC	10	201	1.369e-21	1.417e-6	2.1	1.6	0.5
S298	3[6]	1096[218]	1	22	3	19	C	4	154	1.436e-41	6.493e-4	5.4	5.8	0.7
TBK	6[3]	1569[32]	1	43	18	25	C	1	144	1.223e-55	1.643e-12	1.7	3.2	0.2