Easily Verified IP Watermarking

Anastasios Bikos & Haridimos T. Vergos

Computer Engineering and Informatics Department,
University of Patras, GR 26504, Rio, Greece.
e-mails: {mpikos, vergos}@ceid.upatras.gr

***Abstract*—Several methods of marking designs to help identify ownership have been proposed. All of these methods have a high degree of difficulty in watermark proof. We attack this problem by proposing a new watermarking technique at the state machine level along with assisting hardware to accomplish proof of ownership straightforwardly. Our experiments on benchmark circuits and an example system-on-chip (SoC) indicate a negligible increase in delay and an extremely small increase in the required implementation area.**

I. Introduction

The digital circuits design paradigm has undergone a dramatic change during the last two decades. The original paradigm of having one company developing from original concept through to fabrication has been abandoned in order to cope with the challenges of the emerging fabrication technologies. Nanometer scale feature sizes that allow fabrication of ever more complex but also more expensive chips, have given rise to independent chip foundries that build designs developed by other companies and have shifted the traditional all-purpose monolithic chip design houses to developing custom designs specific to the needs of a part of the total chip.

The current integrated circuits (ICs) are actually Systems on Chip (SoC) composed by multiple components and interconnecting logic and are built in a modular way, pretty much in the same way that a software package is made up by tools written by different coders and vendors. This has led to a new business model where the individual components required in building a SoC are designed by different design houses and are then leased to the SoC developer for a specific project and/or a specific number of chips. Leasing different intellectual property (IP) cores from different design houses and having only to develop their interconnection logic and a very small part of the remaining SoC is the only way for the SoC vendor to meet the strict time to market while in parallel keeping the cost low by having only a small development team.

This IP-based design methodology however, has brought to light a new problem, namely, that of protecting the terms of the IP lease. That is, the IP developer must have a method to verify that his IP is used only for the specific SoC and the number of units for which the royalties have been paid for and must have clear proofs about the ownership of the design in question if in need to litigate against the SoC vendor. This problem has been researched in several other fields that need a proof of ownership, for example in images, video and audio and the most common way to attack it is the use of watermarks [1]. The most common methods and corresponding research efforts for IP watermarking are analyzed in the next section.

In IP watermarking however, there is another problem that is often overlooked. The leased IP is embedded deep in the SoC design hierarchy, making its controllability and observability very low. Therefore, a watermarking method that requires all inputs and all outputs of the leased IP to be controllable and observable for its proof would be hardly of any application to a finished SoC. To this end, in this paper we attack the IP watermarking problem from a different point of view, that of, proving ownership with the least possible available resources. We propose a watermarking method at the state machine level along with accompanying hardware, which can provide proof of ownership using a single input and a single output along with the system clock. We assume that these resources can be made available during the testing of the IP. Our experimental results based on well-known sequential benchmark circuits as well as on a small SoC example, indicate that the proposed method can be implemented with a very small area overhead and with a negligible delay overhead.

The rest of this manuscript is organized as follows. We review the proposed IP watermarking methods in Section II along with a list of their advantages and disadvantages. The proposed watermarking method and its proof of existence mechanism are detailed in Section III. Our experimental method and results are given in IV. We conclude in the last Section.

II. Prior works

The already available IP watermarking methods can be classified into three major categories according to the characteristics of the circuit they are applied. Table I summarizes the pros and cons of the different level methods.

The methods that are applied at the layout level of the circuit (for example [2]–[5]) embed the watermark by either a way similar to paper watermarking, that is, by embedding unused hardware at the layout of the circuit or by imposing several extra constraints at the layout level, such as to place some subfunctions at a certain row or at a certain column or route strangely interconnections that do not belong to the critical path. Small shifts from the standard on-grid placement of specific blocks or unused buried vias also fall in this category of techniques. Since automated place and route tools will remove unused components, will output warning messages for unused ones that can not be removed and will also restore the placement of off-grid components, these methods are well suited only to hard-IPs, that is IPs that the end-user is not allowed to edit. Moreover, proof of ownership can only be claimed by means of microphotographs taken from the finished product.

Another class of methods uses the power signature of the IP. The current drawn by a design for a pair of input vectors depends on the number of transitions caused and can therefore be thought of as a signature for this design for this vector pair. By carefully selecting a series of input vectors, with some of them causing a high number of transitions while the rest only a few the total signature can be considered specific for the particular design and can therefore be used as a watermark [6], [7]. Unfortunately, when an IP is embedded deep in the hierarchy of a SoC, the proof of this signature is very hard, since the vectors that must be applied cause transitions to the rest of the SoC too, while the SoC vendor may be reluctant to provide vectors that put the rest of the IPs of a SoC into a low power state.

Most research efforts however, are focused on IP watermarking solutions that target the finite state machine (FSM) of the sequential circuits [8]–[12]. This category of techniques includes those in which the watermark is inserted as an entry FSM that requires a specific input sequence before the normal operation of the system begins, those that add an authentication FSM in the front end that provides a specific bit sequence when traversed and those that embed a second FSM that generates the watermark output. Several research efforts have attacked the large overhead problem of inserting a completely new FSM by proposing methods for merging part of the watermark FSM with the operation FSM. Such merging may be achieved by adding unused edges for producing a signature on the output that would not normally be produced in the design. For each edge needed if the node has an unused combination it is used to provide the required transition. In the case that no available input sequence exists at any node, the inputs are increased to double the edges from any state. The hardware overhead can be further reduced if the bits in the state encodings are used as substrings of the larger signature that is used as watermark. To this end, recent efforts [13] generate a directed graph representation of the watermark and then attempt to find its best-case graph match with the graph representation of the FSM.

Unfortunately, most of the research efforts in the FSM watermarking category follow a graph-theoretic approach, neglecting the applicability issue and in particular watermark verification. They naively consider that all inputs and outputs, as well as all state encoding bits are available to the IP vendor after the SoC production. This is hardly true however in a real SoC environment, since the inputs of the FSM may come from another IP, while it outputs may drive a third IP. Therefore, for the above methods to be applicable in the modern SoC design paradigm, the resources required for proof of watermark existence must be realistic. To this end, in the next section we describe an IP watermarking method, for which watermark verification can be obtained using the system clock, a single input and a single output.

III. Proposed watermarking method

In this section, we introduce a watermarking method at the state machine level, with a straightforward verification of existence. For exemplifying our method, we consider the FSM of Fig. 1, which describes the Mealy model of a circuit with two inputs and a single output. Our FSM has $k = 7$ states,

TABLE I. Watermarking techniques comparison

Watermarking Class	Pros	Cons
Layout	Implemented easily	Only for hard IPs Verification is expensive
Power	Implemented easily	Verification is hard Vectors must be chosen carefully
State Machine	Implemented easily May provide obfuscation	High overhead possible Possibly easily removed / disabled Questionable ease of verification in SoC environment

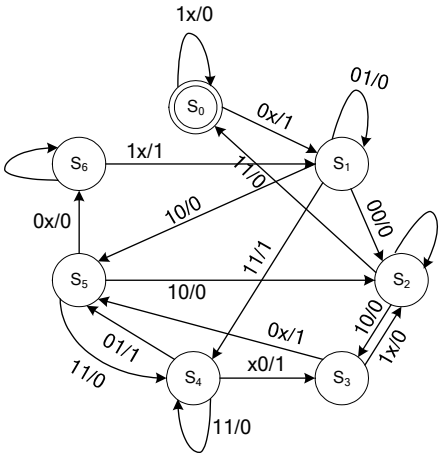

Fig. 1. Original example FSM

with S_0 being the initial state. The following procedures are then applied for attaining a watermarked FSM :

- A set of $m \leq k$ states is selected, along with a random walk over them which starts from the initial state. Since the value of m determines the length of the watermark, a value close to k will produce as wide watermarks as possible. In our example case we select $m = 6$ and the random walk $S_0 \rightarrow S_5 \rightarrow S_4 \rightarrow S_3 \rightarrow S_1 \rightarrow S_6$.
- An extra input, T, is added. All initial edges of the FSM are also present in the watermarked FSM when $T = 0$. Edges for the selected random walk are then added, which have $T = 1$, while all the rest inputs and all outputs are do not care terms.
- We consider a Johnson counter with a length of at least $\lceil \frac{m}{2} \rceil$ cells and a random initial state for it. Its values during $m-1$ consecutive clock cycles are recorded. The string produced during these cycles is our watermark. The m selected states are then encoded according to these values, while the rest states can be encoded arbitrarily. In our example case, for $m = 6$ we consider a 3-bit Johnson counter. Its cells are denoted as $d_2d_1d_0$. If we start from the initial value of $d_2d_1d_0 = 110_2$, then during the 5 next clock cycles we will get the sequence $100, 000, 001, 011, 111$, which gives us the watermark 110100000001011111. Then, the selected states are encodes as follows : $S_0 \rightarrow 110$, $S_5 \rightarrow 100$, $S_4 \rightarrow 000$, $S_3 \rightarrow 001$, $S_1 \rightarrow 011$ and $S_6 \rightarrow 111$. Obviously S_2 which does not belong to our set of selected stages can be encoded arbitrarily.

Fig. 2 presents the watermarked FSM that resulted from the above procedures in our example case.

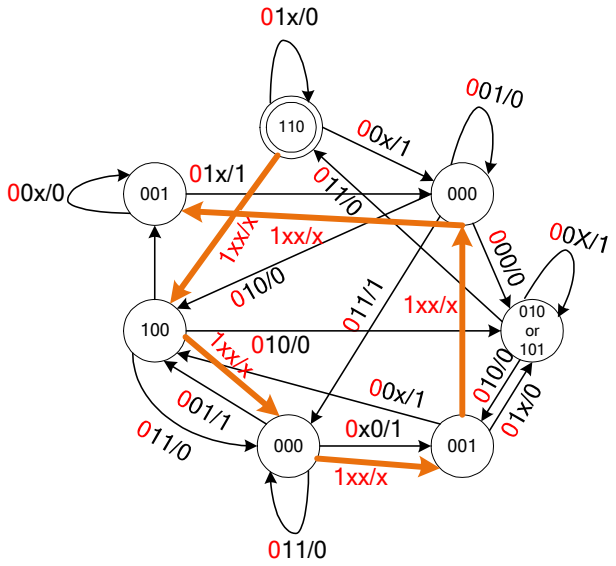

Fig. 2. Watermarked FSM

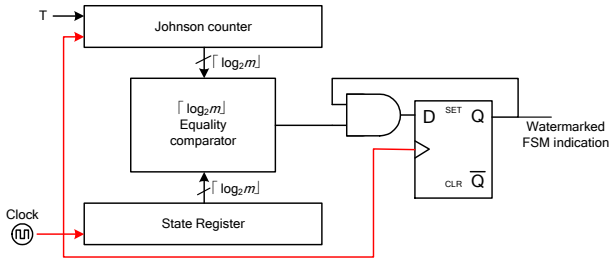

Fig. 3. Extra hardware required for proof of watermark.

Once the FSM has been watermarked as suggested above, watermark verification is straightforward. The following components are embedded in the design to this end :

- a Johnson counter that is initialized to the same value with the encoding of the initial state and accepts T as an enable signal
- an equality comparator that compares the state register value with the value of the Johnson counter, and
- a single flip flop, which is initialized to 1, with its output driven back to its input via a two-input AND gate that also accepts the comparator output.

Fig. 3 presents a block diagram of the hardware embedded to provide ease of watermark proof. For watermark verification, one needs to set T to 1, perform an initialization, so that the initial state of the FSM and the initial value of the Johnson counter are entered. Then, $m-1$ clock cycles are applied. If the watermark is present, this means that during these cycles both inputs of the comparator will be equal and therefore the flip-flop will remain at 1. In case of a different state encoding at any clock cycle, the flip-flop will be cleared and will remain cleared till the end of the $m-1$ clock cycles. At the end of the $m-1$ cycles only the flip-flop output needs to be sampled. An output of 1, indicates that the FSM is watermarked.

In the above discussion we referred to a Johnson counter clearly for keeping the presentation as clear as possible. However, any other hardware structure that can provide the required m state encodings can be used instead, as for example, a linear feedback shift register (LFSR) with a length of at least n, with $n \geq \log_2(m+1)$. An LFSR is a far better solution than a Johnson counter for FSMs with a large number of states, since the required LFSR width increases only logarithmically as m increases.

Obviously, when two or more FSMs are available within the same design, the designer can choose to watermark any number of them for increased security. In this case, the watermarking of more FSMs can be accommodated by only one extra input, that is T. Even with just one FSM present in a design, for increased security, the IP vendor may want to watermark more than one random walks that may have totally distinct or share a number of states. However, in this case, more than one extra input must be added in every edge.

Finally, from the above discussion it becomes clear that in the proposed method there is a clear trade-off between the length of the watermark and the hardware overhead. For a wider watermark more states need to be selected for our random walk, implying a wider Johnson counter and equality comparator. It is also noted that the number of watermarks that can be derived by the proposed method is not simply given by all possible initial states of the Johnson counter. This is because, we can permute the bits of a Johnson counter in any possible way and still get a valid encoding for our states. Of course, the same permutation must be performed in the Johnson counter outputs of Fig. 3 before they are driven to the equality comparator, without any further hardware cost.

IV. Experimental Results

In this section, we evaluate the overhead of the proposed method in terms of area and delay using two different experiments.

For our first experiment we used some of the medium and large sized benchmark circuits (FSMs) in the IWLS 2005 benchmark suite [14]. Starting-off from the HDL description of these circuits we used the Synopsys Design Compiler® tool for synthesizing and mapping them into a 32 nm CMOS implementation technology [15]. A standard synthesis and optimization script from [16] was used, properly modified for targeting the circuit with the least area that can provide an operating frequency of at least 500 MHz. The descriptions of the FSMs were then modified so as to include a watermark according to the proposed method, as well as the extra hardware indicated in Fig. 3 for straightforward watermark proof. For every examined circuit, we considered four different random walks composed of the 25%, 50%, 75% and 90% of the total states. For the smallest of the examined cases (s832) we consider that these states are encoding according to the state sequence of a Johnson counter, while for the rest two according to the states of an LFSR implementing a primitive polynomial. The random walks with 25%, 50%, 75% and 90% of the total states result in 28, 91, 130 and 276 bit watermark for s832, respectively, while for s298 the resulting watermarks are 330, 763, 1312 and 1576 bits wide, respectively. In each case we further considered several initial state encodings. The modified descriptions were then also synthesized and mapped following the same procedure and constraints with the original ones. The attained overhead results averaged over the different initial state encodings are presented in Table II.

TABLE II. OVERHEAD ON BENCHMARK CIRUITS

Circuit	States (k)	Edges	Overhead according to $\frac{m}{k}$ percentage							
			25%		50%		75%		90%	
			Area	Delay	Area	Delay	Area	Delay	Area	Delay
s832	25	245	8.1%	0.18%	11,2%	0.54%	13.4%	0.76%	16.8%	0.83%
s1488	48	251	3.9%	0.26%	6.8%	0.67%	8.9%	0.81%	11.1%	0.92%
s298	218	1096	3.1%	0.39%	5.7%	0.74%	7.8%	0.91%	9.9%	1.07%

The delay overhead on the initial FSM is in all examined cases extremely small; it is in all but one cases less than 1%. Furthermore, it is in all cases imposed by the extra hardware presented in Fig. 3 and not by the insertion of the extra input T and the new edges of the FSM. On the other hand, the area overhead is significant in the two smallest examined circuits (s832 and s1488) especially when the random walk selected is composed by a significant number of states (50% of the total or more). It is less than 10% in all examined cases however, in the largest examined circuit. We should however keep in mind that these results are for circuits that just implement an FSM, whereas an FSM in real life is only a very small part of the IP leased for a complete SoC design.

To this end, in our second experiment we considered a complete SoC built on the AMBA® architecture. The SoC is composed by two AMBA high-performance bus (AHB®) masters, namely a processor and a DMA controller, a bus arbitrer, an AHB to the Advanced Peripheral Bus (APB®) bridging module and an in-house designed AHB slave IP core composed by a 4KB 4-way set-associative write-back cache and its controller. The purpose of the cache is to act as an intermediate between the AHB masters and slower memory, offering zero wait state transactions in most accesses. The AHB protocol supports both single and burst transfers with different address increments of variable widths; namely, bytes, halfwords and words. To cope with all these needs the cache controller contains a medium-sized FSM with 79 states that needs to distinguish the type and size of transfer, in case of a cache hit of a burst transfer to present to the processor the required information and then perform address generation (and possibly wrapping) for the rest of the burst transfer, while in case of a cache miss it needs first to access the missing address from the slow memory imposing the required wait states and then continue with cache block filling. It finally takes care of coherency problems (such as a read access after a write access) and updates the block replacement information after each access. The cache controller along with the cache memory accounts for approximately the 26% of the total SoC logic.

We synthesized the whole SoC without and with the proposed watermarking method in the cache controller FSM, along with the extra watermark verification hardware using a random walk over all the states of the FSM and encoding the states according to the states of an LFSR. The embedded watermark is 553 bits wide. The watermarked SoC offers the exact same clock frequency as the original one, while its area is larger by only 0.12%.

V. CONCLUSIONS

To cope with the ever increasing complexity of today's SoCs, a new business model has been adopted where the individual IP cores required in building the SoC are designed by different design houses and are then leased to the SoC developer for a specific project and / or a specific number of chips. To make sure that the lease terms are followed each IP must be watermarked by its designer. The already proposed watermarking methods at the FSM level totally ignore the fact that the IP will be finally embedded deep in the SoC hierarchy making it too uncontrollable and unobservable for the watermark to be verified.

To this end, in this paper, we have presented a watermarking method along with the required hardware add-ons to simplify watermark verification. Following the proposed method, a single input and a single output only need to be controllable and observable, respectively. Our experimental results indicate that the proposed method can be implemented with negligible delay and small area overheads.

REFERENCES

[1] D. Saha and S. Sur-Kolay, "Robust intellectual property protection of VLSI physical design," *IET Computers Digital Techniques*, vol. 4, no. 5, pp. 388–399, 2010.

[2] E. Charbon, "Hierarchical watermarking in IC design," in *Proceedings of the IEEE 1998 Custom Integrated Circuits Conference*, pp. 295–298.

[3] A. B. Kahng *et al.*, "Watermarking techniques for intellectual property protection," in *Proceedings of the 35th Annual Design Automation Conference*, 1998, pp. 776–781.

[4] T. Nie, "Post layout watermarking design method for IP protection," Ph.D. dissertation, Kochi University, Dept. of Information Science, 2008.

[5] A. Cui, C.-H. Chang, and L. Zhang, "A hybrid watermarking scheme for sequential functions," in *Proceedings of the 2011 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2333–2336.

[6] D. Ziener and J. Teich, "Power signature watermarking of IP cores for FPGAs," *J. Signal Process. Syst.*, vol. 51, no. 1, pp. 123–136, Apr. 2008.

[7] D. Ziener, F. Baueregger, and J. Teich, "Multiplexing methods for power watermarking," in *Proceedings of the 2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 36–41.

[8] A. Abdel-Hamid, S. Tahar, and E. Aboulhamid, "IP watermarking techniques: survey and comparison," in *Proceedings of the 3rd IEEE International Workshop on System-on-Chip for Real-Time Applications*, 2003, pp. 60–65.

[9] ——, "A tool for automatic watermarking of IP designs," in *Proceedings of the 2nd Annual IEEE Northeast Workshop on Circuits and Systems (NEWCAS)*, 2004, pp. 381–384.

[10] ——, "A public-key watermarking technique for IP designs," in *Proceedings of the 2005 Design, Automation and Test in Europe Conference & Exhibition*, 2005, pp. 330–335 Vol. 1.

[11] ——, "Finite state machine IP watermarking: A tutorial," in *Proceedings of the 1st NASA/ESA Conference on Adaptive Hardware and Systems*, 2006, pp. 457–464.

[12] R. Chakraborty and S. Bhunia, "HARPOON: An obfuscation-based SoC design methodology for hardware protection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 10, pp. 1493–1502, 2009.

[13] R. W. P. Meana, "Approximate sub-graph isomorphism for watermarking finite state machine hardware," Master's thesis, University of South Florida, Dept. of Computer Science and Engineering, 2013.

[14] C. Albrecht, "IWLS 2005 benchmarks," in *14th International Workshop on Logic and Synthesis*, 2005.

[15] Synopsys Inc., "Synopsys 32/28nm Generic Library," Available : http://www.synopsys.com/Community/UniversityProgram.

[16] H. Bhatnagar, *Advanced ASIC Chip Synthesis Using Synopsys® Design Compiler®, Physical Compiler® and PrimeTime®*. Springer, 2002.