A blind dynamic fingerprinting technique for sequential circuit intellectual property protection

Zhang, Li; Chang, Chip Hong

2014

Chang, C. H., & Zhang, L. (2014). A Blind Dynamic Fingerprinting Technique for Sequential Circuit Intellectual Property Protection. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 33(1), 76-89.

https://hdl.handle.net/10356/79485

https://doi.org/10.1109/TCAD.2013.2282282

A Blind Dynamic Fingerprinting Technique for Sequential Circuit Intellectual Property Protection

Chip-Hong Chang, *Senior Member, IEEE* and Li Zhang, *Student Member, IEEE*

***Abstract*— Design fingerprinting is a means to trace the illegally redistributed intellectual property (IP) by creating a unique IP instance with a different signature for each user. Existing fingerprinting techniques for hardware IP protection focus on lowering the design effort to create a large number of different IP instances without paying much attention on the ease of fingerprint detection upon IP integration. This paper presents the first dynamic fingerprinting technique on sequential circuit IPs to enable both the owner and legal buyers of an IP embedded in a chip to be readily identified in the field. The proposed fingerprint is an oblivious ownership watermark independently endorsed by each user through a blind signature protocol. Thus the authorship can also be proved through the detection of different user's fingerprints without the need to separately embed an identical IP owner's signature in all fingerprinted instances. The proposed technique is applicable to both application-specific integrated circuit (ASIC) and field-programmable gate array (FPGA) IPs. Our analyses show that the fingerprint is immune to collusion attack and can withstand all perceivable attacks, with a lower probability of removal than state-of-the-art FSM watermarking schemes. The probability of coincidence of a 32-bit fingerprint is in the order of 10^{-10} and up to 10^{35} 32-bit fingerprinted instances can be generated for a small design of 100 flip-flops.**

***Index Terms*— VLSI IPP, fingerprinting, watermarking, field authentication, system-on-chip, synthesis-for-testability**

I. Introduction

Over the last decade and for many years to come, IP-based design methodology has been and will remain a key enabler to improving integrated circuit (IC) design productivity at advanced design processes. According to a recent market research: "The semiconductor IP market is growing in both the IC IP and System-on-Chip (SoC) IP sub-sectors, but the revenues from the SoC IP segment are expected to grow faster at an estimated compound annual growth rate of 19.16% from 2012 to 2017" [1]. This design paradigm has over years dramatically increased the organizational complexity of IC design and pushes for a vertical specialization of organization where stages of IC design are disintegrated, and outsourced to firms and relocated across national boundaries that possess the tacit knowledge and expertise at lower cost. A rising concern of this geographical dispersion of chip design activities is the contaminated chip supply due to the infiltration of counterfeit chips. Electronics Resellers Association International (ERAI), a private group that tracks and fights counterfeit electronics reported that dubious chips are increasing year by year [2]. The perils are alarming when thousands of fake chips from brand names such as Motorola, Intel, Cypress, Altera and National Semiconductors had been sold by a counterfeit-chip broker VisionTech before it was prosecuted in 2010 [3].

Prevailing means of IP protection includes external communications of legal protection, such as patents, copyrights and contracts, to deter illegal IP infringement, and licensing agreement and encryption to deter unauthorized use of IPs. Such approaches, while effective against the benign users, are inadequate to suppress misappropriation of IPs by aggressors, malicious clients, competitors and curious contractors. Owing to the easily accessible, easily integratable and uncommitted physical manifestation nature of reusable IPs, the fraudsters can easily dispute or challenge the validity of the patent or copyright behind an IP, especially if it is not broad, fundamental, or strong with regards to the obviousness test, novelty of idea, or prior art. The lengthy and costly legal process, together with the high degree of uncertainty over the outcome of IP litigation tends to discourage the pursuit of IP infringements. As cases of IP violations by customers are more difficult to prove, IP vendors usually do not take legal action against their customers, resulting in the buoyant IP theft.

As long as infringement remains easy and perpetration is difficult to prove, sizeable portion of the investment in IP development will be extorted. IP providers are in dire need of an effective mechanism to protect not only their ownership rights, but also the legitimacy of the usage of their IPs. Existing watermarking schemes [4-13], though capable of identifying the legal ownership of an IP in case of piracy, is unable to trace the guilty buyer from the unauthorized resold copies of his legally owned IP. This is because the distributed IP instances to different buyers are identical and carry the same mark. This problem can be solved by fingerprinting, which makes the IP instance distributed to each system integrator unique and distinguishable. In this way, when some user illegally redistributes his legally owned IP instance to another party for use in an unauthorized application, the fingerprint embedded in it can be extracted and used to identify the source of misuse.

IP fingerprinting shares much of the desiderata and challenges as IP watermarking, like functionality preservation, high credibility of ownership proof, low design overheads, robustness against removal attacks, transparency to existing design flow and ease of ownership verification. In addition, it

Manuscript received March 22, 2013, revised 15 July 2013

The authors are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: echchang@ntu.edu.sg, lzhang2@e.ntu.edu.sg).

must be able to generate many different high-quality instances with reasonably low design effort. Thus, the intuitive way to fingerprint a design by repeating the entire optimization process of a typical constraint-based watermarking technique [5] to embed a different mark for each user is unacceptable. Very few proposals [14-17] can avoid the complexity of generating a sufficiently large number of quality fingerprinted instances.

The technique proposed in [14, 15] protects FPGA IPs at the physical design level. Fingerprint bits are embedded into the unused logic blocks in unique locations of different tile instances, and FPGA design tiling and partitioning are used to lower the cost of generating many different functionally equivalent circuit instances. As the logic blocks that host the marks are not the only differences between different fingerprinted instances, it is unlikely that tile comparison collusion will succeed in removing a large portion of the fingerprint. To facilitate the recovery of the recipient fingerprint, the owner signature and the recipient fingerprint are always placed in a constant location relative to one another. The reliance on the identification of the locations of ownership marks for the extraction of the correct instance recipient fingerprint may escalate the risk of common ownership mark locations as targets of attack. After all, marks hidden in unused logic blocks can be easily removed without altering the functionality. Moreover, the retrieval of FPGA configuration for mark validation restricts its applicability to mainly static memory based devices. On the other hand, the fingerprinting technique in [16] creates a large solution space by solving the problem only once. The problem solved is not the original problem, but a modified one. From the generated solution, different solutions are derived for the original problem. The general technique is proved by an NP-complete graph coloring problem. For specific IPs, finding the modification to obtain different solutions can be a challenge. The method in [17] solves the problem once to create a seed solution, from which smaller problems are generated. Fingerprinted solutions are then created by re-solving these smaller problems. While the design effort has been greatly reduced, the overhead is still unacceptable when the required number of different IP instances is large. All the above mentioned methods validate the fingerprint by indirect detection. Depending on the method and the level of abstraction where the mark was applied, non-trivial effort may be needed to check if the constraints generated by the fingerprint are satisfied from a deeply integrated IP. In short, none of the existing schemes are able to conveniently detect the embedded fingerprint off-chip after the IP is integrated and packaged as in the dynamic watermarking schemes of [7, 8].

In this paper, a new fingerprinting technique for the protection of sequential circuit IPs is proposed. The method is unique in many ways. It is the first dynamic fingerprinting scheme in which the embedded ownership and buyer's identity of an embedded IP can be conveniently detected off-chip by injecting a specific input sequence. A single fused signature that carries the watermark for ownership proof and fingerprint for buyer identification, as opposed to the concatenation of two separate and independent signatures, is generated through a message exchange in a blind signature protocol. Both the watermark and fingerprint can be detected by running the fingerprinted design or the IC that contains the fingerprinted instance with a specific sequence of input bound to a buyer. The IP provider can easily identify the buyer without divulging any sensitive information about the fingerprint yet the buyer's endorsement of the detected fingerprint is indisputable. These are in stark contrast to other fingerprinting techniques and are made possible by recoding the state variables through the embedding of a test machine into the sequential circuit to be fingerprinted. To the best of our knowledge, this is also the first work that exploits state encoding for fingerprinting sequential circuit IPs. Unlike finite state machine (FSM) watermarking on state transitions or topologies [9-12], which have a tendency to introduce obtrusive dummy inputs, replicated states or redundant transitions, state encoding modifies the state values without introducing new input variable or new state variable and have a global influence on the circuit structure. The fingerprint is thus inherently collusion-resilient as the signatures encoded into the state variables form an integral part of the solution space, and the marked and unmarked circuit components cannot be discriminated by the structural disparities between different fingerprinted instances. To reduce the effort of generating a large number of fingerprinted instances, variable chaining heuristic and dependency-directed partitioning are proposed to break a large sequential circuit into multiple smaller interconnected segments for test machine embedding. The optimized testable segmented circuits are used as seed design for state recoding by the signature. The overheads of different fingerprinted instances have been greatly reduced by the fusion of watermark and fingerprint, the state variable dependency minimization and the pre-optimization of partitioned seed designs. Our method is applicable to sequential circuit IPs targeted for both ASIC and FPGA implementations. As there is no prerequisite of specialized scan flip-flops (FFs) or specific FPGA configuration fabrics, our scheme is transparent and applicable to different technology families of FPGAs. With additional locking and activation mechanism, our fingerprinting technique can also be devised to perform active hardware metering like [18-21]. These schemes can effectively prevent instead of passively confirming IC piracy and chip overproduction by controlling the access to part of the chip's functionality with a specific input sequence. Although schemes [18, 19] embed a watermark into the added obfuscation FSM, the watermark can only authenticate the IP author but not its buyer, and cannot be easily detected off-chip. By combining the active metering technique with our scheme, even if the locking function is cracked, the IP authorship and the buyer responsible for the fraudulent IP instance can still be traced by the fingerprint.

The rest of this paper is organized as follows. Section II introduces the property of test machine and the fundamentals of test machine embedding. The fingerprint generation, embedding and detection of the proposed fingerprinting method are presented in Section III. Section IV analyzes the security of the proposed scheme with respect to its credibility, robustness and feasibility. Experimental results are presented in Section V. The paper is concluded in Section VI.

II. PRELIMINARIES ON TEST MACHINE INSERTION

Our fingerprinting method is founded on the test machine embedding problem of synchronous sequential circuits [22-25]. This section presents the construction of a *n*-FF test machine and explains how its test property can be embedded into an *n*-FF sequential function to synthesize a testable design.

A. Test Machine Fundamentals

An FSM is a quintuple $M = (S, I, O, \Delta, \Lambda)$, where S is a finite set of N states $\{S_1, \ldots, S_N\}$, I is a finite set of p primary inputs $\{y_1, \ldots, y_p\}$, O is a finite set of q primary outputs $\{z_1, \ldots, z_q\}$, $\Delta: S \times I \to S$ is the state transition function and $\Lambda: S \times I \to O$ is the output function [26].

M can be represented by a state transition graph (STG), which is a directed graph whose vertices and edges correspond to the states and state transitions of M respectively; each edge is labeled with the input and output associated with the transition.

An FSM can be tested by checking if its transitions $t_{ij} = (S_i, S_j; y/z)$, $S_i, S_j \in S$, $y \in I$, $z \in O$, $S_j = \Delta(S_i, y)$ and $z = \Lambda(S_i, y)$, are implemented as specified by its STG. This simple conformance test requires a sequence of inputs $y \in I$ to bring the physical FSM from any state to S_i and a sequence of inputs to verify that the FSM reaches the designated state S_j after the stimulating transition t_{ij}.

A *synchronizing sequence* (SS) is defined as an input sequence which, when applied to any initial state of M, will drive M to a single specific state [27]. Likewise, a *distinguishing sequence* (DS) can also be defined as a sequence of inputs such that for any state of M, the sequence of outputs generated in executing that sequence uniquely identifies the state. The output response that uniquely identifies the state is called the *distinguishing response* (DR) of the state.

An *n*-FF test machine is a special FSM, denoted by $M' = (S', I', O', \Delta', \Lambda')$, with $N = 2^n$ states, one input y' and one output z'. Any states $S_i \in S', i \in [0, N-1]$, of it can be set by applying an n-bit SS and be distinguished at the output by applying an n-bit DS. It can be constructed as follows:

$$\Delta'(S_i, y' = a) = \begin{cases} S_{2i} & \text{if } 2i < N \\ S_{2i-N} & \text{otherwise} \end{cases}$$
$$\Delta'(S_i, y' = \bar{a}) = \begin{cases} S_{2i+1} & \text{if } 2i+1 < N \\ S_{2i+1-N} & \text{otherwise} \end{cases} \quad (1)$$

$$\Lambda'(S_i, y') = \begin{cases} b & \text{if } i < N/2 \\ \bar{b} & \text{otherwise} \end{cases} \quad (2)$$

where $a, b \in \{0,1\}$ are the arbitrary constants assigned to the input and output of M'.

Fig. 1 shows the STG for a 3-FF test machine. Let $SS(S_i)$ be the binary representation of SS of state S_i. By convention, the least significant bit (LSB) of $SS(S_i)$ is input first. One important property of the test machine is that the output response is different from each initial state S_i regardless of the input sequence. Hence, any 3-bit sequence can be used as the DS. Let $DR(S_i)$ denotes the binary representation of DR of state S_i such that the LSB of $DR(S_i)$ is output first. The binary representations of SS and DR for the 3-FF machine in Fig. 1 are shown in Table I. If $a = b = 0$, then $SS(S_i)$ and $DR(S_i)$ will be equal to the bit reversal of the binary index i of S_i.

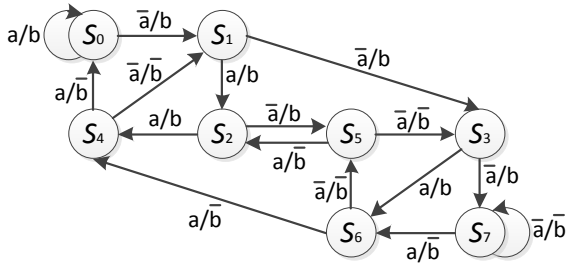


Fig. 1. STG of a 3-FF test machine.

TABLE I
Synchronizing sequences and distinguishing responses of 3-FF test machine

SS	To state	DR							
		S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
aaa	S_0	bbb	$\bar{b}bb$	$b\bar{b}b$	$\bar{b}\bar{b}b$	$bb\bar{b}$	$\bar{b}b\bar{b}$	$b\bar{b}\bar{b}$	$\bar{b}\bar{b}\bar{b}$
$\bar{a}aa$	S_1	bbb	$\bar{b}bb$	$b\bar{b}b$	$\bar{b}\bar{b}b$	$bb\bar{b}$	$\bar{b}b\bar{b}$	$b\bar{b}\bar{b}$	$\bar{b}\bar{b}\bar{b}$
$a\bar{a}a$	S_2	bbb	$\bar{b}bb$	$b\bar{b}b$	$\bar{b}\bar{b}b$	$bb\bar{b}$	$\bar{b}b\bar{b}$	$b\bar{b}\bar{b}$	$\bar{b}\bar{b}\bar{b}$
$\bar{a}\bar{a}a$	S_3	bbb	$\bar{b}bb$	$b\bar{b}b$	$\bar{b}\bar{b}b$	$bb\bar{b}$	$\bar{b}b\bar{b}$	$b\bar{b}\bar{b}$	$\bar{b}\bar{b}\bar{b}$
$aa\bar{a}$	S_4	bbb	$\bar{b}bb$	$b\bar{b}b$	$\bar{b}\bar{b}b$	$bb\bar{b}$	$\bar{b}b\bar{b}$	$b\bar{b}\bar{b}$	$\bar{b}\bar{b}\bar{b}$
$\bar{a}a\bar{a}$	S_5	bbb	$\bar{b}bb$	$b\bar{b}b$	$\bar{b}\bar{b}b$	$bb\bar{b}$	$\bar{b}b\bar{b}$	$b\bar{b}\bar{b}$	$\bar{b}\bar{b}\bar{b}$
$a\bar{a}\bar{a}$	S_6	bbb	$\bar{b}bb$	$b\bar{b}b$	$\bar{b}\bar{b}b$	$bb\bar{b}$	$\bar{b}b\bar{b}$	$b\bar{b}\bar{b}$	$\bar{b}\bar{b}\bar{b}$
$\bar{a}\bar{a}\bar{a}$	S_7	bbb	$\bar{b}bb$	$b\bar{b}b$	$\bar{b}\bar{b}b$	$bb\bar{b}$	$\bar{b}b\bar{b}$	$b\bar{b}\bar{b}$	$\bar{b}\bar{b}\bar{b}$

B. Test machine Insertion

For an arbitrary FSM, due to the limited controllability and observability, it is not always possible to obtain the SS and DS for all the states. These limitations are usually overcome by replacing the FFs of FSM by specialized scan FFs and chaining them to two externally added serial scan input and serial scan output pins. However, such scan insertion procedure is not a useful platform for fingerprinting a sequential design as it is difficult to keep the embedded signature stealthy due to the obtrusive and easily traceable scan FFs and scan IOs. Without introducing specialized scan FFs and additional IO pins, an alternative approach is to insert the test function into the gate-level design of M before logic synthesis by test machine insertion [25].

An *n*-FF test machine $M' = (S', I', O', \Delta', \Lambda')$, can be embedded into an arbitrary *n*-FF FSM $M = (S, I, O, \Delta, \Lambda)$ by isomorphic mapping of the states and IO symbols of M to those of M'. The input y' and output z' of M' can be shared with an input $y \in I$ and an output $z \in O$ of M, respectively by using multiplexers. To share the same set of FFs for multiplexing the next state functions, $\Delta'(S', y')$ and $\Delta(S, y)$, and the output functions, $\Lambda'(S', y')$ and $\Lambda(S, y)$, of M' and M, respectively, the state variables, $x_1', x_2', \cdots, x_n'$, of M' must be mapped to the state variables, $x_1, x_2, \cdots, x_n$, of M. The mapping can be performed in polynomial time for completely specified function without adding new transition to M [24]. If M has unspecified transitions for input y at some state in S, M' may not be compatible to M. To

obtain an isomorphic mapping, a minimum number of additional transitions may have to be introduced into *M* to implicitly specify the responses of some of these unspecified transitions. A new test-enable (*TE*) input can then be introduced into *M* to select between the normal function and the test function.

Let $\pi: X' \rightarrow X$ be the mapping of the set of variables from $X' = \{x'_j\}$ to $X = \{x_i\}$ such that $x_i = \pi(x'_j), \forall i, j \in [0, n-1]$ and $S' = S(\pi(X'))$. Then the next state and output equations of *M* will be modified to

$$X = \Delta'\left(S\left(\pi\left(X'\right)\right), y\right) \cdot TE + \Delta\left(S, y\right) \cdot \overline{TE} \quad (3)$$

$$O = \Lambda'\left(S\left(\pi\left(X'\right)\right), y\right) \cdot TE + \Lambda\left(S, y\right) \cdot \overline{TE} \quad (4)$$

where Δ and Δ' are the next state equations of *X* and *X'* of *M* and *M'*, respectively.

The complexity of the merged next state and output decoders can be reduced by keeping its state variable dependency low in state variable mapping, with the help of a dependency graph (DG). A DG is a directed graph with each vertex representing a FF and each arc representing the signal flow from a FF's output to the same or a different FF's input by abstracting away the combinational logic between them. DG of the object machine *M* can be obtained by tracing the netlist, but DG of the test machine *M'* cannot be obtained before state assignment.

A two-block partition [26] on the state set *S* of *M* divides *S* into two disjoint subsets, i.e., $B_1 \cup B_2 = S$ and $B_1 \cap B_2 = \varnothing$; Each subset of states, $B_1, B_2 \subset S$, is called a block. Let $s_i \equiv s_j$ denote that states s_i and s_j are in the same block of partition. A partition for *M* is said to be closed if $\Delta(s_i, y) \equiv \Delta(s_j, y)$ for all $s_i \equiv s_j$ and $y \in I$, where $s_i, s_j \in S$.

To obtain a state assignment for *M'*, a two-block partition is induced by each state variable such that it is assigned the same value for all states in the same block of the partition. As there are *n* two-block partitions and each block of a partition can be assigned either a '0' or '1' value for its state variable, there are 2^n different state encodings. The dependency of state variables is minimized if the *n* two-block partitions of *M'* are closed [26].

Fig. 2 shows the DG with the least state-variable dependency for the 3-FF test machine with the following state assignment:

FF1': {(S_0, S_2, S_4, S_6), (S_1, S_3, S_5, S_7)},
FF2': {(S_0, S_1, S_4, S_5), (S_2, S_3, S_6, S_7)},
FF3': {(S_0, S_1, S_2, S_3), (S_4, S_5, S_6, S_7)}.

Fig. 2. Minimum dependency graph of 3-FF test machine.

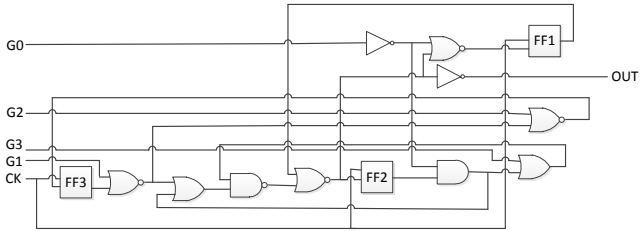


Fig. 3. Gate-level schematic of S27.

As an example, consider the insertion of the 3-FF test machine into the ISCAS'89 benchmark circuit S27 shown in Fig. 3, with its DG extracted in Fig. 4. If (FF1', FF2', FF3') of the test machine is mapped to (FF3, FF1, FF2) or (FF3, FF2, FF1) of S27, there is no increase in dependencies among the state variables.

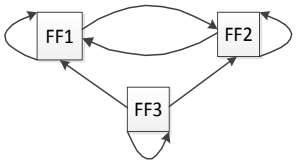


Fig. 4. State variable dependency graph of S27.

Let $a = b = 0$ and $S'(x'_1 x'_2 x'_3)$ of *M'* be encoded as: $S_0(101)$, $S_1(100)$, $S_2(111)$, $S_3(110)$, $S_4(001)$, $S_5(000)$, $S_6(011)$ and $S_7(010)$. From Fig. 1, the next state and output equations of *M'* are given by: $x'^{+}_1 = \overline{y}'$, $x'^{+}_2 = \overline{x'_1}$, $x'^{+}_3 = \overline{x'_2}$ and $z' = \overline{x'_3}$, where x'^{+}_i denotes the next state value of x'_i.

If (x'_1, x'_2, x'_3) are mapped to (x_3, x_2, x_1) of *M*, its state are encoded as: $S_0(101)$, $S_1(001)$, $S_2(111)$, $S_3(011)$, $S_4(100)$, $S_5(000)$, $S_6(110)$, and $S_7(010)$. From Fig. 3, the next state and output equations of *M* are given by: $x_1^{+} = \overline{\overline{G}_0 + \overline{f}_1}$, $x_2^{+} = \overline{f}_1$, $x_3^{+} = \overline{G_2 + f_2}$ and OUT = f_1, where $f_1 = x_1 + \overline{f_3 \cdot f_4}$, $f_2 = \overline{G_1 + x_3}$, $f_3 = G_3 + \overline{G}_0 \cdot x_2$ and $f_4 = \overline{G}_0 \cdot x_2 + f_2$.

As *M* has four input pins, if G_0 is arbitrarily selected to share with the input *y'* of *M'*, then the next state and output equations after test insertion are given by:

$$x_3^{+} = \overline{G}_0 \cdot TE + \overline{G_2 + f_2} \cdot \overline{TE} \quad (5)$$

$$x_2^{+} = \overline{x}_3 \cdot TE + \overline{f}_1 \cdot \overline{TE} \quad (6)$$

$$x_1^{+} = \overline{x}_2 \cdot TE + \overline{\overline{G}_0 + \overline{f}_1} \cdot \overline{TE} \quad (7)$$

$$OUT = \overline{x}_1 \cdot TE + f_1 \cdot \overline{TE} \quad (8)$$

The circuit S27 after test insertion is shown in Fig. 5.

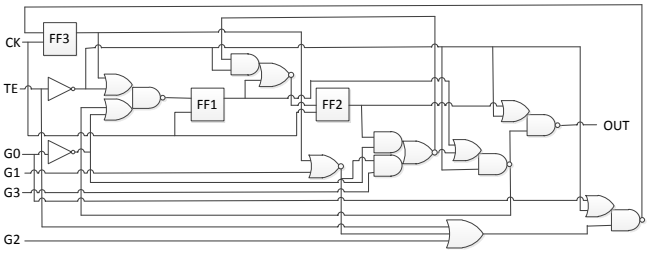


Fig. 5. Schematic of S27 after test machine insertion.

Table II shows the synthesis results of S27 after it has been embedded with each of the 2^3 different encodings of *M'* and resynthesized by Synopsys Design Compiler (DC) using TSMC 0.18 μm technology. Different state encodings of the test machine results in different instances of varying implementation cost. The largest instance is 4.7% larger than the smallest one, the slowest instance is 11.5% slower than the fastest one, and the most power hungry instance consumes 5.1% more power than the least one.

TABLE II
Synthesis results for S27 after test machine insertion

Design	Test machine encoding	Area (μm^2)	Delay (ns)	Power (mW)
1	S_0 (000), S_1 (001), S_2 (010), S_3 (011) S_4 (100), S_5 (101), S_6 (110), S_7 (111)	292.7	0.81	0.267
2	S_0 (100), S_1 (101), S_2 (110), S_3 (111) S_4 (000), S_5 (001), S_6 (010), S_7 (011)	296.1	0.87	0.256
3	S_0 (010), S_1 (011), S_2 (000), S_3 (001) S_4 (110), S_5 (111), S_6 (100), S_7 (101)	286.1	0.81	0.254
4	S_0 (110), S_1(111), S_2 (100), S_3 (101) S_4 (010), S_5 (011), S_6(000), S_7 (001)	289.4	0.82	0.264
5	S_0 (001), S_1 (000), S_2 (011), S_3 (010) S_4 (101), S_5 (100), S_6 (111), S_7 (110)	292.7	0.82	0.266
6	S_0 (101), S_1 (100), S_2 (111), S_3 (110) S_4 (001), S_5 (000), S_6 (011), S_7 (010)	282.7	0.81	0.255
7	S_0 (011), S_1 (010), S_2 (001), S_3 (000) S_4 (111), S_5 (110), S_6 (101), S_7 (100)	289.4	0.78	0.261
8	S_0 (111), S_1 (110), S_2 (101), S_3 (100) S_4 (011), S_5 (010), S_6 (001), S_7 (000)	292.7	0.79	0.265

III. Proposed Fingerprinting Scheme

One unique challenge of IP fingerprinting over IP watermarking is the design effort required to derive a large number of distinct high-quality solutions of the same functionality for different buyers. Another challenge is the ease of recovering the signature of the embedded IP core off-chip without compromising its security against erasure and collusion attacks. Last but not least is the difficulty to lower the area and timing overheads of embedding both the authorship proof and IP buyer's identification in the same design instance. The latter can be addressed by merging the watermark and fingerprint into one single signature but an associated problem is on keeping this signature unobtrusive without losing the non-repudiability of the authorship proof and origin of misappropriation. This section presents a fingerprinting method based on the test machine insertion problem. For ease of exposition, we refer to the original sequential circuit as *object design M* and the sequential circuit after the test machine insertion as *testable design* M_T. The testable design with the best synthesis result in terms of area or timing is chosen as the *seed design* M_S for fingerprinting.

A. Fingerprint Generation

Instead of routinely generating the watermark and fingerprint as two independent signatures, our scheme uses a unitary signature to provide an undeniable proof of IP ownership and for traceability of illegal IP distribution. This signature is generated through a blind signature protocol [28] to preserve the anonymity of the IP authorship information endorsed by the buyer. The blind signature protocol requires a digital signature mechanism and a commuting function. For simplicity, we consider Chaum's blind signature protocol based on RSA. Let (n_A, e_A) and d_A be the certified RSA public and private keys of Buyer *A*, where $n_A = p_A \cdot q_A$ is the product of two large random primes. The fingerprint *F* to be embedded into the IP instance for Buyer *A* can be generated through the following message exchange between the IP provider and Buyer *A*.

1. *Watermark generation*: The IP provider selects a message to convey its ownership information. The ASCII encoded message is converted into a binary string and reduced by a message digest (MD) such as SHA-2 [29] to an integer *W*, where $0 \le W < n_A$.
2. *Blinding phase*: When Buyer *A* makes a purchase request, the IP provider randomly selects a secret integer k_A satisfying $0 \le k_A < n_A$ and $\gcd(n_A, k_A) = 1$ to compute $W_A = W \cdot (k_A)^{e_A} \bmod n_A$. This concealed watermark W_A is then sent to Buyer *A*.
3. *Signing phase*: To endorse the purchase, Buyer *A* signs W_A with his secret key d_A and returns his blinded signature $F_A = (W_A)^{d_A} \bmod n_A$ to the IP provider.
4. *Unblinding phase*: By computing $F = F_A \cdot k_A^{-1} \bmod n_A$, the IP provider obtains the signature of Buyer *A* on his watermark *W* , which is the fingerprint to be inserted into the IP instance sold to Buyer *A*.

The IP provider can verify if the fingerprint *F* is genuine by decrypting it with Buyer *A*'s public key. Since *F* is the digital signature of Buyer *A* on the IP provider's watermark *W*, it provides an undeniable proof for tracing the redistribution of the copies of IP bought by him. Meantime, as *W* is concealed in W_A, Buyer *A* has no knowledge of the IP provider's watermark *W* and his own signature on *W*. The blindness of *F* increases the deterrent effect of uncertainty.

B. Fingerprint Insertion

By applying automatic test pattern generation on the seed design, a set of combinational test vectors can be obtained. Each test vector is then converted to a test sequence which includes an SS to excite a testable design to a specific state S_s at test mode (i.e., *TE* = '1') and a parallel input stimulus to produce the output of a transition from S_s to some destination state S_d at capture mode (i.e., *TE* = '0'). Irrespective of the input stimulus, the destination state S_d from any starting state S_s can always be identified by its DR based on the test machine property. As demonstrated in Table I for $n = 3$, the DR of any destination state can be observed from the primary output *z* by applying an arbitrary input sequence of length *n* through the primary input *y* at test mode.

To insert the fingerprint $F = \{f_i\}_{i=0}^{m-1}$ into an object design, a test vector *V* for the seed design M_S is randomly selected. Let S_s and S_d be the initial and destination states of M_S for the test vector *V*. To randomly modulate *m* out of *n* ($m \ll n$) binary bits of $DR(S_d)$ by *F*, a keyed one-way pseudorandom number generator (PNG) is used to generate an order set $L = \{l_i\}_{i=0}^{m-1}$, of *m* unique integers between 0 and *n*–1 such that $l_i \in [0, n-1]$, $\forall i = 0, \cdots, m-1$ and $l_i \ne l_j, \forall i \ne j$. l_i corresponds to the location of f_i, where f_0 is the LSB of *F*. The keyed one-way PNG can be realized by SHA-2 or any other cryptographically secure hash function, as long as it is computationally infeasible to find a collision of the same group of numbers without the knowledge of the secret key.

According to *L*, a fingerprint-compatible $DR^* = \{r_j\}_{j=0}^{n-1}$ is generated by setting $r_j = f_i$ if $j = l_i$ for all $i = 0, \cdots, m-1$ and r_j ='–' otherwise, where '–' denotes a don't-care value.

A binary ({0, 1}) sequence A is said to be compatible to a ternary ({0, 1, −}) sequence B of the same length, denoted as $A \subset B$, if all but the don't care values of A and B in the same bit positions are matched, i.e., $a_i = b_i, \forall a_i \in A, b_i \in B$ and $b_i \neq$'–'. If $DR(S_d)$ is compatible with DR^*, M_S will be used as the fingerprinted instance M^*. Otherwise, a subset $S^* = \{S_u \in S \mid DR(S_u) \subset DR^*\}$ is extracted from the state set S of M_S such that the DRs of all its members are compatible with DR^*. The fingerprinted instance can then be generated by modifying the state encoding of M_S by $\phi: x_i \rightarrow \dot{x}_i, \forall i = 1, \cdots, n$, where $\dot{x}_i = x_i$ or $\bar{x}_i$, such that the encoded value of one of its states $S_d^* \in S^*$ is equal to the encoded value of S_d, i.e., $S_d^* = \phi(S_d)$. This recoding of state variables can be performed through the embedded test machine of M_S. If there are more than one state encodings that satisfy this requirement, the testable design instance with the least overhead is selected as the fingerprinted design M^*. A fingerprint verification code V^* can then be derived from V by replacing its $SS(S_s)$ by $SS(S_s^*)$, where $S_s^* = \phi(S_s)$. The fingerprint insertion process is depicted in Fig. 6.

```
fingerprint(M_seed, F, L, V)
{
   M_seed: Seed design of n FFs;
   F: Fingerprint to be inserted, binary vector of length m;
   L: List of m location indices;
   V: Randomly selected combinational test vector;
   r: Ternary vector of length n
   for (j = 0; j < n; j++) {
      r[j] = '–';
      for (i = 0; i < m; i++)
         if (j == L[i]) {
            r[j] = F[i];
            break;
         }
   } // end for loop to obtain r
   if (DR(S_d) ⊂ r) return (M_seed, V); // if DR is compatible with r
   CS = ∅; // create an empty list of DR-compatible states
   for (i = 0; i < 2^n; i++) // search for DR-compatible states
      if (DR(S_i) ⊂ r) CS += S_i;
   min_overhead = MAX_VALUE; // initialize minimum overhead
   for each( 2^n state encoding φ of M) {
      if (φ(S_d) ∈ CS) {
         fingerprint_M = resynthesize(M_seed, φ);
         overhead = overhead(fingerprint_M, criteria)
         if (overhead < min_overhead) {
               min_overhead = overhead;
               opt_M = fingerprint_M;
               opt_V = modified_SS(V, φ(S_s)); // modify SS of V
         }
      }
   } // end search for optimal fingerprinted instance
   return (opt_M, opt_V);
}
```

Fig. 6. Fingerprint insertion algorithm.

As an example, consider the insertion of a one-bit signature $f_0 =$'1' into the 3-FF testable design of S27. If design area is the prime concern, the smallest Design 6 in Table II is selected as the seed design M_S. Let $V =$"110;0100;– – –" be a randomly selected test vector. The first 3-bit vector is $SS(S_3)$, the 4-bit vector is applied to the primary inputs "$G_0G_1G_2G_3$" at capture mode to bring $S_s = S_3$ to $S_d = S_4$ and the 3-bit don't care vector is the DS to obtain $DR(S_4)$. For $a = b = 0$, $DR(S_4) =$"001" from Table I. If L = {1}, then $DR^* =$"–1–". Since $DR(S_4)$ is not compatible with DR^*, a search for all states S_i with $DR(S_i) \subset DR^*$ is performed. From Table I, the compatible states are $S^* = \{S_2, S_3, S_6, S_7\}$. From Table II, it can be observed that the encoded values of S_2 in Design 7, S_3 in Design 3, S_6 in Design 8 and S_7 in Design 4 are mapped to the encoded value of S_4 = "001" of M_S. Among them, Design 3 has the least area and is selected as the fingerprinted design M^*, with $\phi: x_i \rightarrow \bar{x}_i$, $S_s^* = \phi(S_3) = S_4$ and $S_d^* = \phi(S_4) = S_3$. From Table I, $SS(S_s^*)$ = "001" and $DR(S_d^*)$ = "110". Thus, $V^* =$"001;0100;– – –". By applying V^* to M^*, f_0 = '1' can be detected from the second bit (i.e., l_0 = 1) of DR^*.

C. Partitioning for Efficient Fingerprinting

It is difficult to obtain an optimized seed design for fingerprinting by resynthesis when the number of FFs n is large. The resynthesis can be made more efficient by partitioning the object design into smaller interconnected segments. If each segment contains only c FFs, it becomes affordable to synthesize all the 2^c testable design instances to obtain an optimized seed design for each segment if $c << n$. The total number of resynthesis processes is reduced to $\lfloor n/c \rfloor \cdot 2^c + |n|_c \cdot 2^{|n|_c}$, where $\lfloor \bullet \rfloor$ is the floor function and $|a|_b$ denotes the remainder of a divided by b, and only a very small part of the design is resynthesized in each process.

As the DG of the test machine has a chain-like structure as shown in Fig. 2, multiple c-FF test machines can be easily cascaded to form a linear chain of state variables. To minimize the dependencies added in the DGs of the partitioned design, the longest chain removal technique [24] is used to chain the state variables of the object design before it is partitioned into smaller interconnected segments. The result is an ordered list of FFs that can be used to direct the partitioning of an n-FF object design M. The following procedure outlines the *state variable chaining heuristic*.

(1) Extract the DG of M by omitting the self-dependencies of the FFs.
(2) Compute strongly connected components (SCCs)[1] of the DG and replace each SCC in the DG with a single vertex to form a directed acyclic graph (DAG).
(3) Obtain the longest chain of the DAG.
(4) Obtain the longest chain in each SCC.
(5) Replace the vertices representing the SCCs in the chain obtained in Step (3) with the chains obtained in Step (4) to form the *longest chain* of the DG.

[1] A SCC of a directed graph $G(V, E)$ is a maximal set of vertices $C \subseteq V$ such that for every pair of vertices $\{u, v\} \in C$, there is a directed path from u to v and a directed path from v to u.

(6) Remove the vertices in the current longest chain and their associated edges from the DG. Repeat Steps (1) to (5) for the new DG. The newly extracted longest chain obtained in Step (5) is then prefixed to the existing longest chain.
(7) Repeat Step (6) until there is no vertex left in the DG.

To find the longest chain for a directed graph containing loops is an *NP*-complete problem [30]. Therefore, loops are removed in Step (2) to simplify the DG to DAG so that dynamic programming [31] with linear computational complexity can be used to extract the longest chain from the DAG. Finding the longest chain in a SCC is also an *NP*-complete problem. This problem in Step (4) is simplified by determining the longest chain from the spanning tree of SCC. The DG is reduced by pruning its longest chain in Step (6). The process repeats until no vertex is left in the DG. A linear chain of state variables for the object design is obtained with dependencies added merely for the connections from the tail of a newly extracted longest chain to the head of an existing chain between two iterations.

The algorithm for the partitioning of the object design into *c*-FF segments is shown in Fig. 7. The output list of each segment after the partitioning is recorded. The output list of segment *i* contains the outputs of FFs or gates in segment *i* that are either primary outputs or inputs to FFs or gates in other segments.

```
dependency_directed_partitioning(M, FF, c)
{
  M: Sequential design to be partitioned;
  FF: List of FFs of M ordered according to state variable chain;
  Gate: List of gates of M;
  IO: List of primary inputs and primary outputs of M;
  c: Predefined maximum number of FFs in each partition;
  Gate[i].used = 0; FF[i].used = 0; // Initialize all gates and FFs of M
  j = 0;
  while (FF is not empty) {
    Partition[j] = create_partition(); //create a new partition
    if (number of unused FFs in FF > c)
      Partition[j].FF += last c unused FFs of FF;
    else
      Partition[j].FF += all unused FFs of FF;
    mark_used_FFs(); // mark those FFs added to Partition as used;
    for each (Partition[j].element) { // element can be Gate or FF
      Predecessor = get_predecessor(Partition[j].element);
      if (Predecessor == FF[k] and FF[k] ∉ Partition[j].FF)
          Partition[j].input += FF[k].output;
      else if (Predecessor == Gate[k])
        if (Gate[k].used == 0){  // Gate[k] is unused
          Partition[j].gate += Gate[k];
          Gate[k].used = 1;
        } else  // Gate[k] is used
            if (Gate[k] ∉ Partition[j].Gate)
              Partition[j].input += Gate[k].output;
      else  Partition[j].input += IO[k]; // Predecessor is IO[k]
    }  //end for loop
    j++;
  }  //end while loop
  return Partition;
}
```

Fig. 7. Dependency-directed partitioning algorithm.

The object design M is partitioned into t smaller interconnected segments, M_0, M_1, ..., M_{t-1}, each with n_i FFs, where $i \in [0, t-1]$ and M_0 denotes the segment that is nearest to the output z. Then an n_i-FF test machine is inserted into each M_i independently as described in Section II with 2^{n_i} possible state encodings. These 2^{n_i} different testable designs for M_i are resynthesized and the most optimized testable design is selected as the seed design M_{Si} for M_i.

To disperse the fingerprint $F = \{f_j\}_{j=0}^{m-1}$ with $L = \{l_i\}_{i=0}^{m-1}$ location indices into the partitioned seed design, the fingerprint-compatible DR^* is divided into t ternary strings, $DR_i^* = \{r_{i,j}\}_{j=0}^{n_i-1}$ for $i = 0, \ldots, t-1$ and $r_{i,j} \in \{0, 1, '-'\}$. DR_i^* can be generated by setting $r_{i,j} = f_k$ if $\sum_{b=0}^{i-1} n_b + j = l_k$, $\forall k = 0, \cdots, m-1$ and $r_{i,j} = '-'$ otherwise. If all bits of DR_i^* are don't cares, no fingerprint bit needs to be inserted into M_{Si} and the fingerprinted instance $M_i^* \equiv M_{Si}$. Otherwise, the fingerprinting algorithm in Fig. 6 is used to insert the fingerprint bits in DR_i^* to produce the fingerprinted instance M_i^*. All fingerprinted instances M_i^* are then cascaded back together and resynthesized to produce the final fingerprinted design M^*.

As an example, consider the ISCAS benchmark S344. Its DG after removing the self-dependencies is shown in Fig. 8, where the FF dependencies are represented by solid directed edges. Using the state variable chaining heuristic, the existing longest FF chain $FF1 \rightarrow FF2 \rightarrow FF3 \rightarrow FF15 \rightarrow FF4 \rightarrow FF5 \rightarrow FF6 \rightarrow FF7 \rightarrow FF8 \rightarrow FF9 \rightarrow FF10 \rightarrow FF11$ is first extracted. After removing the FFs in the longest chain from the DG, *FF*14, *FF*13 and *FF*12 are individually extracted as the longest chains in the subsequent iterations. By appending the existing chain to the newly extracted longest chains, the final chain of FFs is given by $FF12 \rightarrow FF13 \rightarrow FF14 \rightarrow FF1 \rightarrow FF2 \rightarrow FF3 \rightarrow FF15 \rightarrow FF4 \rightarrow FF5 \rightarrow FF6 \rightarrow FF7 \rightarrow FF8 \rightarrow FF9 \rightarrow FF10 \rightarrow FF11$. New dependencies are added to form this FF chain, which are indicated by dashed directed edges in Fig. 8.

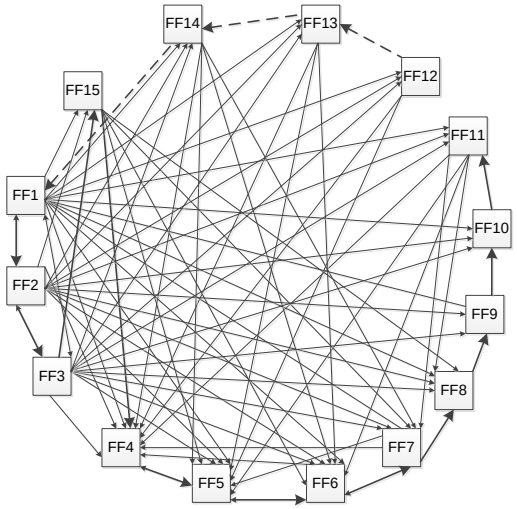


Fig. 8. Dependency graph of S344 excluding self-dependencies. Solid edges are the existing dependencies; dark and dashed edges are respectively the selected and added dependencies of the final FF chain.

Assume the maximum number of FFs per segment is 3, the algorithm in Fig. 7 returns the following partitions:

M_0 with {x_3(*FF*11), x_2(*FF*10), x_1(*FF*9)},
M_1 with {x_3(*FF*8), x_2(*FF*7), x_1(*FF*6)},
M_2 with {x_3(*FF*5), x_2(*FF*4), x_1(*FF*15)},
M_3 with {x_3(*FF*3), x_2(*FF*2), x_1(*FF*1)},
M_4 with {x_3(*FF*14), x_2(*FF*13), x_1(*FF*12)}.

Table III shows the synthesized areas for each partitioned testable design M_{Ti}. Designs 1 to 8 correspond to the 8 different state encodings of the embedded 3-FF test machine in the second column of Table II. The embedding is performed by mapping (x_1', x_2', x_3') of the test machine to (x_1, x_2, x_3) of each segment M_i. The best designs, Design 6 for M_{S0}, Design 3 for M_{S1}, Design 5 for M_{S2}, Design 1 for M_{S3} and M_{S4} are selected as the seed designs for fingerprinting.

TABLE III
Synthesis results (area in μm^2) for test machine embedded components of S344

Design	M_0	M_1	M_2	M_3	M_4
1	439.1	575.5	459.0	319.3	282.7
2	415.8	595.4	485.7	349.3	316.0
3	439.1	568.8	452.4	319.3	292.7
4	415.8	602.1	479.0	349.3	312.7
5	439.1	575.5	445.7	319.3	282.7
6	409.1	595.4	472.3	342.6	316.0
7	439.1	568.8	452.4	319.3	292.7
8	415.8	602.1	479.0	349.3	312.7

Assume a fingerprint F = "110001" and its location indices L = {5, 11, 9, 2, 4, 0}. Then DR^* = "−−−1−0−−−10−0−1" is the fingerprint compatible DR. It can be partitioned into DR_0^* = "0−1", DR_1^* = "10−", DR_2^* = "−−−", DR_3^* = "1−0" and DR_4^* = "−−−". Since DR_2^* = DR_4^* = "−−−", M_{S2} and M_{S4} can be used as fingerprinted instances M_2^* and M_4^*, respectively. Let V = "011110001011001; 001010111; −−−−−−−−−−−−−−−−" be the test vector selected for the interconnected seed designs $M_{S4} \rightarrow M_{S3} \rightarrow M_{S2} \rightarrow M_{S1} \rightarrow M_{S0}$ with S_s = "S_6-S_3-S_4-S_6-S_4", S_d = "S_7-S_2-S_7-S_7-S_4" and $DR(S_d)$ = "111<u>0</u>1011111<u>1</u>1001". The underlined bits of $DR(S_d)$ are incompatible with DR^*. Since DR_0 = "001" ⊂ "0−1", M_{S0} can be used as the fingerprinted instance M_0^*. DR_1 = "111" ⊄ "10−". From Table I, for $a = b = 0$, $DR(S_1)$ = "100" and $DR(S_5)$ = "101" are compatible with "10−". From Table II, the encoded value of S_5 in Design 1 and S_1 in Design 2 can be mapped to the encoded value of $S_d = S_7$ = "101" of M_{S1}. Design 1 has lower area than Design 2 of M_1, and is selected as M_1^*, then $\phi: (x_1, x_2, x_3) \rightarrow (x_1, \bar{x}_2, x_3)$, and $S_s^* = \phi(S_6) = S_4$ and $S_d^* = \phi(S_7) = S_5$. From Table I, $SS(S_4)$ = "001" and $DR(S_5)$ = "101". Similarly, DR_3 = "010" ⊄ "1−0". $DR(S_1)$ = "100" and $DR(S_3)$ = "110" are compatible with "1−0". The encoded value of S_1 in Design 7 and S_3 in Design 5 can be mapped to the encoded value of $S_d = S_2$ = "010" of M_{S3}. Both designs have the same cost and either of them can be selected as the fingerprinted instance M_3^*. If Design 7 is selected, then $\phi: (x_1, x_2, x_3) \rightarrow (x_1, \bar{x}_2, \bar{x}_3)$, and $S_s^* = \phi(S_3) = S_0$ and $S_d^* = \phi(S_2) = S_1$. From Table I, $SS(S_0)$ = "000" and $DR(S_1)$ = "100". The fingerprinted instances can be chained to form $M_{S4} \rightarrow M_3^* \rightarrow M_{S2} \rightarrow M_1^* \rightarrow M_{S0}$ and resynthesized to produce the fingerprinted design M^*. Hence, V^* = "011<u>000</u>001<u>001</u>001; 001010111; −−−−−−−−−−−−−−−−" and DR = "111<u>100</u>111<u>101</u>001", where the underlined bits are the modified values of S_s^* and S_d^* of M_1^* and M_3^* after fingerprinting. The area, delay and power consumption of the seed and final fingerprinted instances are (1616.6 μm^2, 1.19 ns, 1.13 mW) and (1623.3 μm^2, 1.18 ns, 1.18 mW), respectively.

D. Fingerprint Detection and Authorship Verification

The IP provider keeps a safe repository of records. Each record consists of the fingerprint F, the secret integer k used in the blinding phase, the buyer's public key (n, e), the fingerprint location vector L and the test vectors V^* for each fingerprinted instance. To detect the authorship of a fingerprinted design, the IP provider can apply the test vector V^* to obtain a DR, from which the fingerprint F can be recovered at the bit locations specified by L. The buyer's public key (n, e) corresponding to F can then be retrieved from the database. The IP provider's watermark can be detected by $W = F^e \bmod n$. The IP provider can prove his ownership of the IP by demonstrating that his legible ownership message can be hashed to W. Since the IP provider's watermark W can be successfully recovered by decrypting the fingerprint F with the specific IP buyer's public key, it proves that the fingerprinted design is sold to that buyer. Thus the buyer of a fingerprinted design can be tracked and held responsible if the fingerprinted design was found to be illegally copied and redistributed.

Decrypting F to detect W needs only to be done once but to detect F from a fingerprinted instance of unknown buyer may require the application of all the different test vectors V^*. To aid the detection of F, the IP provider can store the responses DR_a to a common test vector V_a for all fingerprinted instances into the corresponding entries of the database. With high probability, DR_a for the same test vector V_a will be different for different instances. By applying V_a to the instance, the number of test vectors V^* required to detect F will be reduced dramatically to only those few matching records of DR_a.

E. Augmented Fingerprint

To increase the attackers' difficulty to successfully erase the fingerprint F, error correction code (ECC) can be added to it. More fingerprint bits will have to be modified in order to prevent F from being correctly detected by the IP provider. Owing to the randomized dispersion by L, the ECC bits are automatically interleaved with the fingerprint bits in DR. If necessary, F can be compressed by cryptographic hash function to adapt to the embedding capacity of the design. The IP provider needs only to demonstrate additionally that F can be hashed to the embedded bit stream recovered from the sold design.

IV. Security Analysis of Proposed Scheme

No hardware IP protection scheme is complete without an analysis of its *credibility*, *robustness* and *feasibility*. Credibility refers to the strength of ownership proof, robustness refers to the resistance against known and perceivable attacks, and feasibility refers to the effort in finding the number of quality

solutions required for the protection scheme. These attributes are analyzed below.

A. Fingerprint Credibility

A common measure to quantify and compare the credibility or strength of a fingerprinted solution is the probability of coincidence (P_c), which are the odds that a non-fingerprinted design carries the fingerprint by coincidence. A lower P_c implies a stronger proof of ownership and original recipient. In our scheme, since each bit of DR of a design is equally probable to be '0' or '1', the probability that m randomly selected bits in DR match a specific binary string of fingerprint is given by:

$$P_c = \frac{1}{2^m} \tag{9}$$

With a 64-bit fingerprint, the P_c of a fingerprinted instance is as low as 5.4×10^{-20}. This probability reduces to the order of 10^{-40} for a 128-bit fingerprint.

B. Fingerprint Robustness

The following attack scenarios are analyzed with Alice as the IP provider using our fingerprinting scheme to protect her IP core and Bob as a malefactor attempting to steal her IP.

(1) *Collusion attack*. This attack requires a coalition of multiple copies of fingerprinted instances in possession by one or more malicious users in order to create a forged copy without the fingerprint for redistribution. Bob may collude with other buyers to compare multiple protected instances to find out their structural dissimilarities, with the hope that these dissimilar parts of the circuits can be modified at an acceptable cost and effort to remove the fingerprint without losing the usefulness and correct functionality of the IP. To succeed in such an attack, the fingerprint must be altered to an extent that none of the users in the coalition can be identified by Alice.

While this attack is a biggest threat to data hiding algorithms on multimedia content in which the hidden signature (watermark or fingerprint) does not depend on the host data, several features in our fingerprinted instance make such attack ineffective. First, the fingerprint bits randomly modulate the values of state variables that affect the next state and output decoding logic in both the normal and test functions, making them inextricable from the functional components of the IP. This makes it very challenging for Bob to figure out the functionality of any structural mismatch in order to successfully remove even a single fingerprint bit. Secondly, the same segment M_i^* of different fingerprinted instances can be identical (i.e., both equal to M_{Si}) or different irrespective of the value and number of fingerprint bits embedded in them. Lastly and most importantly, the entire chain of testable machines is resynthesized as a whole to obtain M^*. The domino effect of the hard optimization problem involved in the resynthesis process whitens the differences between the fingerprint-modulated and unmodulated subcircuits, causing the entropy of identifying and associating the matching or mismatching subcircuits between any two instances to increase with the number of fingerprinted instances being compared.

(2) *Combinational logic resynthesis*. Bob may attempt to remove the fingerprint by performing a combinational logic resynthesis through various approaches [32]. Although this attack can successfully modify the circuit structure by changing the combinational logic implementations without affecting the functionality, the input and output behaviors of the sequential elements are preserved. Thus, Alice can still recover the fingerprint F from the DR of the fingerprinted design modified by Bob. One characteristic of the test machine is that it can be decomposed into multiple smaller test machines and independently embedded into a correspondingly partitioned large machine. Although the original circuit topologies of these interconnected machines may not be maintained upon global optimization of the final testable design, the full controllability and observability of all FFs are not affected. As an example, consider the STGs of one-FF and two-FF test machines shown in Fig. 9. If they are interconnected such that the output of the first test machine is the input of the second test machine, then the $2^1 \times 2^2 = 8$ combined states can be mapped to the 8 states for the 3-FF test machine of Fig. 1. Let the state of the final test machine be represented as $(S_{1,i}, S_{2,j})$ when the first test machine is in state $S_{1,i}$ and the second test machine is in state $S_{2,j}$. If the bijection maps $(S_{1,i}, S_{2,j})$ to S_k of the 3-FF test machine, then $SS((S_{1,i}, S_{2,j}))$ and $DR((S_{1,i}, S_{2,j}))$ of the combined machine can still be determined from $SS(S_k)$ and $DR(S_k)$, respectively in Table I for the 3-FF test machine. Our method utilizes this property to efficiently distribute the fingerprint bits and propagate the embedded fingerprint F to all subsequent stages of the design flow. The fact that $SS(S_k)$ and $DR(S_k)$ remain intact after the resynthesis of the final testable design implies that F will survive all forms of logic synthesis and optimization performed on the fingerprinted instance.

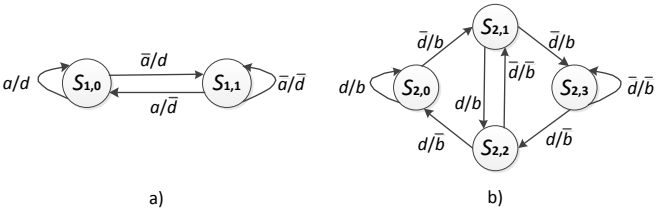


Fig. 9. STGs of (a) 1-FF and (b) 2-FF test machines.

(3) *Circuit retiming*. Circuit retiming relocates the FFs of sequential circuit to improve its performance while preserving the functionality. Bob may retime the fingerprinted instance. As retiming may change the STG of the circuit, it seems possible for Bob to partially remove the fingerprint inserted by Alice if the altered parts of the STG host the fingerprint bits.

According to [33], retiming merges two states that are one-step equivalent or splits one existing state into two new states that are one-step equivalent. Two states are said to be one-step equivalent if they have the same output and the same destination state for any input. Fortunately, each state of the test machine is distinct and the fingerprinted instance contains no one-step equivalent state. Hence, none of the states in the fingerprinted instance can be merged by circuit retiming. Meantime, as the test machine is fully specified, it is also not possible to split one state of the fingerprinted instance into two one-step equivalent states without adding extra sequential elements. Even if Bob manages to create one-step equivalent states at the cost of additional circuitry without losing the usefulness of Alice's IP, the states and their associated

transitions involved in the fingerprint extraction are retained. Thus Alice can still recover the correct fingerprint from the DR of the retimed fingerprinted instance redistributed by Bob.

(4) *State recoding*. There are two possible ways by which Bob can recode the state variables of Alice's protected design. The first way is to extract the STG directly from the fingerprinted design to carry out the state recoding. It has been proven that extracting the STG from a large sequential circuit after logic synthesis is computationally intractable [9]. Thus this approach is infeasible in practice. Alternatively, Bob may perform a partitioning on the fingerprinted instance to obtain many smaller segments of the design. With no knowledge of the design or architecture of Alice's IP, Bob may only be able to extract a limited number of STGs for some segments to recode their state variables. Since the buyer has no knowledge of the embedded fingerprint F and input vector V^* for its recovery, Bob can only select the segments by wild guess. Due to the existence of loops (SCCs) in the dependency graph, the solution for the chaining of state variables with minimum additional dependencies is not unique. Various new SCCs have been formed after the embedding of test machines, and the final resynthesis of the entire fingerprinted design further masks the partitioning and chaining of state variables. Hence, it is highly unlikely that Bob could precisely relate the fingerprint bits to the corresponding state variables. It is reasonable to assume that Bob is incapable of extracting the STG for every partition, recode and re-synthesize the STG to forge a completely new design as this task incurs no less effort than the complete redesign of the IP functionality with a high, and almost certain risk of violating the circuit functionality or performance. Even if he succeeds in forging a new design, he will not be able to generate the proper test vectors for the counterfeit design without knowing the encodings used by the test machines, which will devalue his design.

(5) *State reduction*. IP protection schemes, such as [9], that leverage on redundancy to mark the states to display some specific property for authorship proof are susceptible to this kind of attack. This vulnerability does not exist in our fingerprinting scheme because the embedded test machine is a fully specified machine with distinct states. The final fingerprinted design contains no redundant state as, if any, it would have been reduced upon the resynthesis performed after fingerprint insertion. In addition, state reduction is usually performed on explicit STG [34], which implies a very high cost of attack on a fingerprinted design instance released at the gate level or lower level of design abstraction.

C. Fingerprint Feasibility

Generally, the feasibility of a fingerprinting scheme is evaluated by three main criteria. First, the solution space required for the generation of a large number of unique instances. Second, the additional cost and effort required to produce a fingerprinted instance. Preferably, the fingerprinting method is compatible with existing design tools and no new investment of tool is needed to embed or detect the fingerprint. The additional effort required should also be substantially lower than designing the IP from scratch. Finally, the impact on quality degradation of any fingerprinted instance should be minimized and kept within a tolerable limit.

The first criterion can be evaluated by the number of possible solutions to fingerprint an IP. The number of combinations to select m state variables from a design of n FFs to host the m fingerprint bits is given by:

$$ {}^{n}C_{m} = \frac{n!}{m!(n-m)!} \tag{10} $$

As each state variable can be assigned either a value of '0' or '1', the number of different fingerprinted instances that can be generated by our scheme is:

$$ 2^{m} \cdot {}^{n}C_{m} = \frac{2^{m} \cdot n!}{m!(n-m)!} \tag{11} $$

For n = 100 and m = 32, the solution space has exceeded 6×10^{35}.

For the second criterion, the fingerprint embedding process of our scheme is completely transparent and can be used with existing design tools and standard cell libraries for ASIC implementation. As no specialized scan FF is required, the scheme is also applicable to fingerprint IPs for different configuration technologies of FPGAs, including the antifuse- or flash-memory based FPGAs for which [15] may not be applicable. The additional design effort to generate each fingerprinted design instance is marginal after the seed design is obtained. Only those segments that are not compatible with its corresponding fingerprinted sequence DR^* at the fingerprint locations specified by L need to be remapped and a final re-synthesis is needed to globally optimize the entire fingerprinted design after the segments have been restitched.

Lastly, our fingerprinting scheme incurs insignificant overhead on the design quality. This is attributed to the efficient dependency-directed partitioning algorithm and the utilization of seed designs, which are optimized for each partitioned testable machine. The low fingerprinting overhead will be demonstrated by experiments on different benchmark circuits in the next section.

V. Experimental Results

Sequential circuits from ISCAS'89 benchmark suite are used to analyze the quality of the fingerprinted instances of our scheme. Depending on the circuit size, the first 16, 32, 64, 128 or 256 bits of the keyed SHA-2 hash value of the fingerprint F were inserted. The location integers L were generated with a SHA-2 based PNG. The design partitioning, test machine embedding as well as the fingerprint insertion algorithms were coded in C++. Combinational test generation on the seed design was carried out with DfT tool DFTadvisor and ATPG tool Fastscan from Mentor Graphics. As part of the fingerprint insertion algorithm, the selected combinational test pattern is converted into a sequential test pattern of the form {SS; test vector; DR}. The synthesis was performed using Synopsis DC with TSMC 0.18 µm technology library, with C-Shell scripts written to automate the synthesis processes for the encoding instances of each segment as well as the whole design.

The synthesis results are shown in Table IV. The area and delay were simulated by Synopsys DC, and the power by Synopsys PrimeTime PX with back annotation. The area, delay and power overheads due to fingerprint insertion are on average

1.6%, 1.2% and 3.4% respectively, and below 2.5%, 5.0% and 5.0% respectively for most circuits.

TABLE IV
Fingerprinting overheads with state-encoding for area reduction. *#FF* is the number of FFs; A_s and A_m, D_s and D_m, and P_s and P_m are areas in μm^2, delays in ns and powers in mW of the non-fingerprinted optimized seed design and fingerprinted design, respectively; ΔA, ΔD and ΔP are the respective percentage increases.

Circuit	*#FF*	A_s	D_s	P_s	m	A_m	D_m	P_m	ΔA (%)	ΔD (%)	ΔP (%)
S382	21	2212	1.35	1.26	16	2229	1.42	1.33	0.77	5.19	5.32
S838	32	3323	3.60	1.82	16	3453	3.31	1.71	3.91	–8.06	–6.04
S1423	74	8452	5.39	5.10	16	8472	5.38	5.42	0.24	–0.19	6.28
					32	8482	5.03	5.34	0.35	–6.68	4.67
S5378	179	17347	1.68	12.0	32	17414	1.65	12.5	0.39	–1.79	4.17
					64	17540	1.78	12.6	1.11	5.95	5.00
S9234	211	24143	2.67	14.3	32	24462	2.22	15.0	1.32	–16.85	4.90
					64	24702	2.8	14.2	2.32	4.87	–0.70
S15850	534	54151	4.04	34.5	64	54862	3.96	38.8	1.31	–1.98	12.46
					128	55473	4.04	35.3	2.44	0.00	2.32
S13207	638	56496	3.18	35.5	64	57068	3.76	36.2	1.01	18.24	1.97
					128	57633	3.16	36.1	2.01	–0.63	1.69
S35854	1426	159274	3.44	95.5	128	161234	3.57	98.1	1.23	3.78	2.72
					256	163206	3.60	103.8	2.47	4.65	8.69
S38417	1636	166097	3.40	67.7	128	167677	3.78	67.7	0.95	11.18	0.00
					256	168625	3.67	68.8	1.52	7.94	1.62
S35932	1728	150506	1.49	112	128	153370	1.51	110.5	1.90	1.34	–1.34
					256	155752	1.52	120.9	3.49	2.01	7.95

Let $R_F = m/n$ represents the fingerprinting ratio, which is the number of fingerprint bits to the number of bits in DR. The area overhead increases with R_F for all designs. This is expected as the originally optimized encoding instance will have to be recoded in more segments as R_F increases. However, the overhead of fingerprinting one design with a smaller R_F may not necessarily be lower than that of another design with a larger R_F. This could be due to the design and topology dependent merging of functional and test logic of different encoding instances. For some circuits, the differences in area, delay and power among different encoding instances are very small. The fingerprinting overheads of these designs may only increase mildly with R_F as opposed to those that have a large quality gap between a suboptimal and the most optimized encoding instances.

As area is used as the selection criterion for the seed design, the delay and power overheads have wider spreads than the area overhead among different designs. For instance, some fingerprinted designs are more than 18% slower than their seed designs while some other could be about 17% faster. This inconsistency is because a smaller instance may not necessarily be faster or consume less power. Hence, the fingerprinted segment that uses a suboptimal encoding instance may have shorter delay or consumes less power than the optimal encoding instance. This explains why the delays or power consumptions of some fingerprinted designs in Table IV are smaller than their seed designs.

The choice of state encoding influences the complexity of the logic functions of the FSM and hence the area, timing and power dissipation of the synthesis results. Thus, the policy for selecting the optimal state encoding can be adapted to the optimization target. Table V shows the synthesis results using area-delay product as the selection criterion if the area and delay of the fingerprinted design are equally important. This selection policy reduces the delays of fingerprinted instances from those of Table IV by 0.17% on average but their areas are also correspondingly increased by 0.12% on average.

TABLE V
Fingerprinting overheads with state encoding for area-delay product reduction. The definitions and units of all notations are the same as those in Table IV.

Circuit	*#FF*	A_s	D_s	P_s	m	A_m	D_m	P_m	ΔA (%)	ΔD (%)	ΔP (%)
S382	21	2202	1.42	1.27	16	2265	1.49	1.37	2.86	4.93	8.21
S838	32	3400	3.32	1.93	16	3463	3.42	1.89	1.85	3.01	–1.92
S1423	74	8436	5.23	5.01	16	8416	4.80	5.10	–0.24	–8.22	1.86
					32	8422	4.84	5.46	–0.17	–7.46	9.05
S5378	179	17064	1.68	12.4	32	17114	1.69	12.6	0.29	0.60	1.61
					64	17264	1.64	12.8	1.17	–2.38	3.23
S9234	211	24160	2.51	15.2	32	24409	2.32	15.8	1.03	–7.57	3.95
					64	24868	2.33	15.8	2.93	–7.17	3.95
S15850	534	54217	4.17	35.9	64	54736	4.06	37.3	0.96	–2.64	3.90
					128	55071	4.07	38.6	1.58	–2.40	7.52
S13207	638	57074	3.62	36.9	64	57869	3.41	38.5	1.39	–5.80	4.34
					128	57909	3.67	38.0	1.46	1.38	2.98
S35854	1426	160306	3.61	97.0	128	161686	3.70	98.0	0.86	2.49	1.03
					256	163523	3.60	97.0	2.01	–0.28	0.00
S38417	1636	166184	3.79	71.6	128	168017	3.46	72.7	0.80	–8.71	1.54
					256	169430	3.93	72.0	1.65	3.69	0.56
S35932	1728	152559	1.56	120	128	155473	1.58	120	1.91	1.28	–0.58
					256	157339	1.58	125	3.13	1.28	3.99

In order to evaluate the area and timing overheads of our fingerprinted circuits on FPGA, designs in Table IV are implemented on Xilinx XC6VCX240T device. The results are shown in Table VI. The number of slice registers used by each design is the same as *#FF* in Table IV. For most designs, similar trend of increasing overhead with R_F is observed. On average, the area and delay overheads are 1.8% and 1.4% respectively. Unlike the fingerprinted solutions of [15], the area and delay of our fingerprinted solutions can be improved as manifested by the negative ΔA and ΔD values in Table VI. This is possible because the resynthesis after state recoding may perturb the optimization heuristic to the FPGA fabric mapping, placement and routing problems to escape the local minima. The fingerprinted design may be easier to be placed and routed than the seed design, leading to the use of less logic slices or shorter interconnections for nearly half of the fingerprinted solutions. In addition, our technique can also be applied to IPs targeting antifuse and flash memory based FPGA technologies, which are not feasible for [15]. Due to the dissimilar approaches to logic minimization, the optimal encodings for FPGA may be different from ASIC. The impact of encoding and decoding logic on power consumption is usually higher in FPGA than in ASIC, particularly for large FSMs. Among the 2^n different minimal-bit encodings, Gray encoding and those with reduced hamming distance of the most probable state transitions are more likely to lead to lower resource utilization and power consumption.

Reported experimental results of fingerprinting methods [14-17] based on only one or a few fingerprint instances for each design are inadequate to demonstrate their capability to generate a large number of reasonably high-quality fingerprinted designs. To evaluate this capability, the same experimental settings for obtaining the data in Table IV were used to synthesize 100 different fingerprinted instances with

randomly generated fingerprints and location integers for each design. The minimum, maximum and average areas and delays of the 100 fingerprinted designs for each circuit are listed in Table VII. The minute margin of error percentages, $\%A_{err}$ and $\%D_{err}$, indicate that different fingerprinted instances of the same IP produced by our technique have consistent quality. By comparing the mean area and mean delay in Table VII with the seed design area A_s and delay D_s in Table IV, a more informative group average fingerprinting overheads, ΔA and ΔD, of 1.4% and 2.3% respectively are obtained.

TABLE VI
Area and timing overheads of fingerprinted designs implemented on FPGA. A_s and A_m, and D_s and D_m are the areas in number of slice LUTs and delays in ns of the seed design and fingerprinted design, respectively.

Circuit	A_s	D_s	m	A_m	D_m	ΔA (%)	ΔD (%)
S382	43	2.58	16	41	2.34	–4.65	–9.44
S838	57	2.61	16	60	2.79	5.26	7.18
S1423	174	4.99	16	176	5.60	1.15	12.19
			32	166	5.51	–4.60	10.85
S5378	223	3.54	32	232	3.79	4.04	7.01
			64	230	3.42	3.14	–3.25
S9234	422	4.16	32	421	5.24	–0.24	25.83
			64	422	4.60	–0.00	10.48
S15850	784	6.05	64	799	5.43	1.91	–10.31
			128	853	5.39	8.80	–10.89
S13207	709	6.04	64	698	5.68	1.27	–5.98
			128	718	5.83	1.91	–3.48
S35854	2660	6.43	128	2669	6.48	0.34	0.82
			256	2674	6.36	0.53	–1.09
S38417	2565	6.94	128	2730	6.97	6.43	0.42
			256	2720	6.96	6.04	0.27
S35932	1729	3.58	128	1749	3.57	1.16	–0.31
			256	1775	3.38	2.66	–5.67

TABLE VII
Statistical analysis of synthesis results for 100 different fingerprinted instances of each circuit. A_m and D_m denote area in μm^2 and delay in ns, respectively. $\%A_{err}$ and $\%D_{err}$ are obtained by calculating the margins of error A_{err} and D_{err} for the area and delay for 95% confidence interval and normalized them by the average area and delay, respectively.

Circuit	m	A_m			D_m			$\%A_{err}$	$\%D_{err}$
		min	Ave.	max	min	Ave.	max		
S382	16	2179	2240	2305	1.32	1.44	1.56	0.26	0.68
S838	16	3320	3388	3473	3.02	3.37	3.78	0.18	0.85
S1423	16	8369	8472	8589	4.71	5.25	5.82	0.12	1.05
	32	8363	8473	8575	4.71	5.28	5.85	0.10	0.98
s5378	32	17349	17358	17550	1.61	1.76	2.06	0.08	0.84
	64	17361	17395	17653	1.62	1.78	1.98	0.11	0.76
s9234	32	24166	24422	24662	2.22	2.51	2.92	0.09	1.01
	64	24266	24579	24915	2.23	2.49	2.87	0.12	0.95
s15850	64	54540	54851	55271	3.42	3.95	4.21	0.05	0.75
	128	54999	55327	55674	3.48	3.99	4.45	0.05	0.79
s13207	64	56788	57094	57397	3.00	3.46	3.90	0.04	0.95
	128	57227	57529	57936	3.16	3.52	4.01	0.05	0.87
s35854	128	160605	161356	161976	3.12	3.57	4.20	0.03	0.96
	256	162122	162880	163729	3.12	3.64	4.11	0.04	1.01
s38417	128	166819	167573	168299	3.41	3.67	4.14	0.03	0.82
	256	168199	168859	169430	3.41	3.75	4.32	0.03	0.91
s35932	128	152905	153419	154109	1.43	1.50	1.64	0.03	0.56
	256	154980	155678	156743	1.43	1.53	1.64	0.04	0.61

To the best of our knowledge, no FSM fingerprinting technique has been reported so far. Among the FSM watermarking schemes, [8] is the only technique that deals explicitly with the problem of field authentication (or off-chip detection) of embedded sequential circuit IP ownership. To facilitate the comparison of our technique with [8], the same synthesis tool SIS [35], technology library *msu.genlib*, optimization script *script.algebraic* and ISCAS benchmark circuits in *blif* format as [8] are used to produce our fingerprinted circuit results in Table VIII. In most cases, our fingerprinted designs have lower area than the watermarked designs of [8], with an average area reduction of about 2% even though reduction of hardware overhead has been well acknowledged as a more challenging problem in fingerprinting than in watermarking. Fingerprinting by state variable encoding of the embedded partitioned test machine has not only resulted in better area optimization but also provided a stronger integration of test and functional logic than the scan-chain based synthesis-for-testability technique. The delay comparison is inconclusive however. Due to the cost function used by the algebraic script from SIS, its metaheuristic tends to create a larger delay discrepancy between the originally optimized cost surface and the alternate cost surfaces than Synopsys design compiler.

TABLE VIII
Comparison of synthesis results with FSM watermarking [8]. A_s and A_m, and D_s and D_m are the areas in μm^2 and delays in ns of the seed design and fingerprinted design, respectively.

Circuit	#FF	m	A_s	D_s	A_m		D_m	
					Prop.	[8]	Prop.	[8]
S3330	132	32	29696	45	30024	33504	49.8	46.4
		64			30088	33496	54.2	47
S3384	183	32	42704	69.6	42920	47736	85	83.2
		64			43440	46160	88	83.6
S9234	228	32	57824	67.6	58752	59528	74.2	65.6
		64			58944	59520	73.8	65.2
S6669	239	32	69720	142.8	70728	75136	126.2	134.6
		64			72608	75312	127	134.2
S15850	597	64	129184	132.6	130568	130928	157.2	123
		128			132280	130880	170.4	123.6
S13207	669	64	118632	128.4	120904	126024	143.8	140.2
		128			123920	123456	154.6	131
S38584	1452	64	338816	469.8	343424	345872	484.2	400.6
		128			346608	342520	490.6	401.6
S38417	1636	64	389320	387	391768	372288	404.4	354
		128			393776	373696	423.4	356.4
S35932	1728	64	330968	354.8	333712	366528	361.2	348.2
		128			335328	366920	363	347
B14	245	32	137512	143.6	137896	139664	171.8	142.4
		64			138296	139176	175.4	142.4
B15_1	449	64	220610	132.4	222208	219152	139.4	147.6
		128			223112	218208	148	148
B21	490	64	287080	154.2	288184	298464	196.6	155.2
		128			288880	298720	207.6	155.2
B22	735	64	428008	215.4	429448	429832	274.4	210.2
		128			430760	430168	272.4	209.6
B17	1415	64	710784	374.4	710960	690296	382.4	404.6
		128			712160	689840	403.4	403

We also compared the areas and delays of our fingerprinted circuits from Table V with those of [9] for the same signature lengths of 64 and 128 bits. The percentage area and delay overheads for each circuit are shown in Table IX. Overall, our scheme incurs smaller area and delay overheads. Moreover, being essentially a watermarking scheme, [8] and [9] are incapable of generating many high-quality topologically different designs (for different marks) of the same IP.

Table IX
Comparison of area and delay overhead with [9]

Circuit	*m*	Δ*A* (%)		Δ*D* (%)	
		proposed	[9]	proposed	[9]
S5378	64	1.17	2.50	–2.38	–7.50
S9234	64	2.93	2.97	–7.17	–1.50
S15850	64	0.96	0.86	–2.64	–3.70
	128	1.58	2.06	–2.40	–2.28
S13207	64	1.39	3.40	–5.80	–3.38
	128	1.46	5.15	1.38	–3.38
S38584	128	0.86	0.75	2.49	4.76
S38417	128	0.80	–5.17	–8.71	–2.99
S35932	128	1.91	–5.02	1.28	2.35

The probability of successful removal of fingerprint bits by state recoding is also analyzed. Based on the resilience analysis of Section IV.B, Bob is able to recode only a small number *r* ($r \ll n$) of state variables from a few randomly created partitions. The probability that the *r* state variables he recoded had already been embedded with exactly *j* fingerprint bits is given by $\left({}^{m}C_{j} \cdot {}^{n-m}C_{r-j}\right) \Big/ {}^{n}C_{r}$, and the probability of modifying exactly *i* bit values by recoding a *j*-bit binary code is given by ${}^{j}C_{i}\left(\tfrac{1}{2}\right)^{i}\left(\tfrac{1}{2}\right)^{j-i} = {}^{j}C_{i}\left(\tfrac{1}{2}\right)^{j}$. Thus, the probability of successfully removing *k* ($k \le m$) or more fingerprint bits by randomly recoding *r* out of *n* state variables is given by:

$$P_r(E \ge k) = \sum_{j=k}^{m}\left(\frac{{}^{m}C_{j} \cdot {}^{n-m}C_{r-j}}{{}^{n}C_{r}} \sum_{i=k}^{j} \frac{{}^{j}C_{i}}{2^{j}}\right) \tag{12}$$

Table X compares the robustness of the proposed method with [8] by the probability of successfully erasing a quarter of the mark, assuming conservatively that the attacker is able to recode $r = m$ state variables for our method and randomly derange $r = m$ FFs for [8]. It can be seen that the proposed method is generally more robust against removal attacks than [8]. As we assume $r = m$, P_r is relatively high when R_F is large. When R_F is around 0.1, P_r reduces drastically to the order of 10^{-8}. The probability of removal is expected to be reduced across-the-board if $r < m$. This observation suggests that it is unlikely for state recoding attack to succeed in altering a large fraction of the fingerprint bits even with a moderate R_F. With ECC, it is highly improbable to remove even a single bit of fingerprint for a reasonable-size design. Given a minimum fingerprint length m_{min} for an allowable probability of coincidence and an empirically determined R_{Fmax} for an allowable probability of erasure, the smallest IPs should contain no less than $n_{\max}$ FFs, where

$$n_{\max} = \frac{m_{\min}}{R_{F_{\max}}} \tag{13}$$

VI. Conclusion

A new and robust fingerprinting technique on sequential circuits that offers a convenient way to identify the legal IP owner and users has been proposed. A large number of high quality fingerprinted instances can be created from an originally optimized seed design of the IP with incremental design effort. The overheads incurred by different instances have been effectively bounded to a reasonably low level by our method due to its state variable chaining heuristic, dependency based partitioning algorithm and the use of a single blind signature to double as IP ownership and buyer identity marks. Our security analysis shows that the fingerprint credibility increases exponentially with fingerprint length and is immune to collusion attack, circuit resynthesis, retiming and state reduction attacks. The risk of fingerprint removal by state recoding attack is significantly reduced with the augmentation of ECC or by keeping the ratio of fingerprint length to state variable number low. From 100 fingerprinted instances for each of the ten ISCAS benchmark circuits tested, the average area and delay overheads were found to be 1.4% and 2.3% respectively with negligible margins of error for 95% confident interval.

Table X
Comparison of mark robustness with [8]

Circuit	*#FF*	*m*	$P_r(E \ge m/4)$	
			proposed	[8]
S3330	132	32	2.6E–02	3.1E–02
S3384	183	32	3.5E–03	3.1E–02
S9234	228	32	8.1E–04	3.3E–02
S6669	239	32	5.8E–04	3.9E–02
S15850	597	64	5.2E–08	5.6E–08
S13207	669	64	1.0E–08	1.1E–08
S38584	1452	64	9.8E–14	1.0E–13
S38417	1636	64	1.6E–14	1.7E–14
S35932	1728	64	6.8E–15	7.0E–15
B14	245	32	4.9E–04	3.3E–02
B15_1	449	64	2.7E–06	3.4E–03
B21	490	64	8.1E–07	3.6E–03
B22	735	64	2.6E–09	2.8E–09
B17	1415	64	1.5E–13	1.5E–13

References

[1] Market research report, "Semiconductor (silicon) Intellectual Property (IP) market (2012 – 2017) global forecasts and analysis by form factor, design architecture, processor type, applications and geography," *MarketsandMarkets*. [Online]. Available: *http://www.marketsandmarkets.com/Market-Reports/semiconductor-silicon-intellectual-property-ip-market-651.html*.

[2] C. Gorman, "Counterfeit chips on the rise," *IEEE Spectrum*, vol. 49, no. 6, pp. 16-17, June 2012.

[3] B. Rayner, "Chip counterfeiting case exposes defense supply chain flaw," *EE Times News & Analysis*, 24 Oct. 2011.

[4] I. Hong, and M. Potkonjak, "Techniques for intellectual property protection of DSP designs," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, WA., USA, May 1998, vol. 5, pp. 3133-3136.

[5] A. B. Kahng *et al.*, "Constraint-based watermarking techniques for design IP protection," *IEEE Trans. Comput.-Aided Des. Integr. Circuits and Systs*, vol. 20, no. 10, pp. 1236-1252, Oct. 2001.

[6] A. Cui, C. H. Chang, and S. Tahar, "IP watermarking using incremental technology mapping at logic synthesis level," *IEEE Trans. Comput.-Aided Des. Integr. Circuits and Systs*, vol. 27, no. 9, pp. 1565-1570, Sep. 2008.

[7] Y. C. Fan, "Testing-based watermarking techniques for intellectual-property identification in SOC design," *IEEE Trans. on Instrumentation and Measurement*, vol. 57, no. 3, pp. 467-479, Mar, 2008.

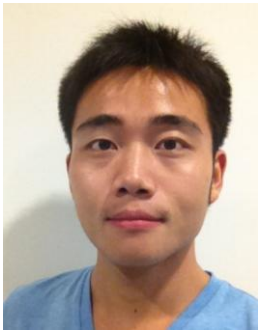
[8] C. H. Chang, and A. Cui, "Synthesis-for-testability watermarking for field authentication of VLSI intellectual property," *IEEE Trans. Circuits Syst. I-Regular Papers*, vol. 57, no. 7, pp. 1618-1630, Jul. 2010.

[9] A. L. Oliveira, "Techniques for the creation of digital watermarks in sequential circuit designs," *IEEE Trans. Comput.-Aided Des. Integr. Circuits and Systs*, vol. 20, no. 9, pp. 1101-1117, Sep. 2001.

[10] I. Torunoglu, and E. Charbon, "Watermarking-based copyright protection of sequential functions," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 3, pp. 434-440, Mar. 2000.
[11] A. T. Abdel-Hamid, S. Tahar, and E. M. Aboulhamid, "A public-key watermarking technique for IP designs," in *Proc. Design, Autom. Test Europe*, Munich, Germany, Mar. 2005, vol. 1, pp. 330-335.
[12] A. Cui *et al.*, "A robust FSM watermarking scheme for IP protection of sequential circuit design," *IEEE Trans. Comput.-Aided Des. Integr. Circuits and Systs*, vol. 30, no. 5, pp. 678-690, May 2011.
[13] L. Zhang, and C. H. Chang, "State encoding watermarking for field authentication of sequential circuit intellectual property." in *Proc. IEEE Int. Symp. on Circuits and Syst.*, Seoul, Korea, May 2012, pp. 3013-3016.
[14] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "FPGA fingerprinting techniques for protecting intellectual property," in *Proc. IEEE Custom Integr. Circuits Conf.*, CA, USA, May 1998, pp. 299-302.
[15] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "Fingerprinting techniques for field-programmable gate array intellectual property protection," *IEEE Trans. Comput.-Aided Des. Integr. Circuits and Systs*, vol. 20, no. 10, pp. 1253-1261, Oct. 2001.
[16] G. Qu, and M. Potkonjak, "Fingerprinting intellectual property using constraint-addition," in *Proc. Des. Autom. Conf.*, CA, USA, June 2000, pp. 587-592.
[17] A. E. Caldwell *et al.*, "Effective iteractive techniques for fingerprinting design IP," *IEEE Trans. Comput.-Aided Des. Integr. Circuits and Systs*, vol. 23, no. 2, pp. 208-215, Feb. 2004.
[18] R. S. Chakraborty, and S. Bhunia, "Hardware protection and authentication through netlist level obfuscation," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, CA, USA, Nov. 2008, pp. 674-677.
[19] R. S. Chakraborty, and S. Bhunia, "HARPOON: An obfuscation-based SoC design methodology for hardware protection," *IEEE Trans. Comput.-Aided Des. Integr. Circuits and Systs*, vol. 28, no. 10, pp. 1493-1502, Oct, 2009.
[20] F. Koushanfar, Y. Alkabani, "Active control and digital rights management of integrated circuit IP cores," in *Proc. Int. Conf. on Compilers, architectures, and synthesis for embedded systems.*, GA, USA, Oct. 2008, pp. 227-234.
[21] F. Koushanfar, "Provably Secure Active IC Metering Techniques for Piracy Avoidance and Digital Rights Management," *IEEE Trans. on Information Forensics and Security*, vol. 7, no. 1, pp. 51-63, Feb. 2012.
[22] V. Agrawal, and K.-T. Cheng, "Finite state machine synthesis with embedded test function," *Journal of Electronic Testing*, vol. 1, no. 3, pp. 221-228, Oct. 1990.
[23] S. Kanjilal, S. T. Chakradhar, and V. D. Agrawal, "A test function architecture for interconnected finite state machines," in *Proc. Int. Conf. on VLSI Design*, Kolkata, India, Jan. 1994, pp. 113-116.
[24] S. Kanjilal, S. T. Chakradhar, and V. D. Agrawal, "Test function embedding algorithms with application to interconnected finite-state machines," *IEEE Trans. Comput.-Aided Des. Integr. Circuits and Systs*, vol. 14, no. 9, pp. 1115-1127, Sep. 1995.
[25] S. Kanjilal, S. T. Chakradhar, and V. D. Agrawal, "A partition and resynthesis approach to testable design of large circuits," *IEEE Trans. Comput.-Aided Des. Integr. Circuits and Systs*, vol. 14, no. 10, pp. 1268-1276, Oct. 1995.
[26] Z. Kohavi, and N. K. Jha, *Switching and Finite Automata Theory*. 3rd ed., New York: Cambridge University Press, 2010.
[27] C. Pixley, S. W. Jeong, and G. D. Hachtel, "Exact calculation of synchronization sequences based on binary decision diagrams," in *Proc. ACM/IEEE Des. Autom. Conf.*, CA, USA, June 1992, pp. 620-623.
[28] D. Chaum, "Blind signatures for untraceable payments," in *Advances in Cryptology*, D. Chaum, R.L. Rivest, & A.T. Sherman Eds., Plenum, 1982, pp. 199-203.
[29] National Institute of Standards and Technology (NIST). Secure Hash Standard (SHS) (March 2012). [Online]. Available: http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf.
[30] M. R. Garey, and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1990.
[31] T. H. Cormen, *Introduction to Algorithms*. 3rd ed., Cambridge, Mass.: MIT Press, 2009.
[32] G. D. Hachtel, and F. Somenzi, *Logic Synthesis and Verification Algorithms*, New York: Springer, 2006.
[33] R. K. Ranjan *et al.*, "On the optimization power of retiming and resynthesis transformations," in *Proc. ACM/IEEE Int. Conf. Comput.-Aided Des.*, CA, USA, Nov. 1998, pp. 402-407.
[34] J. K. Rho *et al.*, "Exact and heuristic algorithms for the minimization of incompletely specified state machines," *IEEE Trans. Comput.-Aided Des. Integr. Circuits and Systs*, vol. 13, no. 2, pp. 167-177, Feb. 1994.
[35] E. M. Sentovich *et al.*, "Sequential circuit design using synthesis and optimization," in *Proc. IEEE Int. Conf. Comput. Des.: VLSI in Comput. and Processors*, MA, USA, Oct. 1992, pp. 328-333.

Chip-Hong Chang (S'92–M'98–SM'03) received the B.Eng. (Hons.) degree from the National University of Singapore in 1989, and the M. Eng. and Ph.D. degrees from Nanyang Technological University (NTU), Singapore, in 1993 and 1998, respectively. He served as a Technical Consultant in industry prior to joining the School of Electrical and Electronic Engineering (EEE), NTU, in 1999, where he is currently an Associate Professor. He holds joint appointments with the university as Assistant Chair of Alumni of the School of EEE since June 2008, Deputy Director of the Center for High Performance Embedded Systems from 2000 to 2011, and the Program Director of the Center for Integrated Circuits and Systems from 2003 to 2009. He has coedited one book, published four book chapters and around 200 research papers in refereed international journals and conferences. His current research interests include hardware security and trust, low power arithmetic circuits and digital filter design.

Dr. Chang has served as Associate Editor of *IEEE Access* since 2013, *IEEE Transactions on Circuits and Systems-I* from 2010-2013, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* since 2011 and *Integration, the VLSI Journal* since 2013, and the Editorial Advisory Board Member of *Open Electrical and Electronic Engineering Journal* since 2007 and *Journal of Electrical and Computer Engineering* since 2008. He also guest edited several special issues for *Journal of Circuits, Systems and Computers*, *IEEE Transactions on Circuits and Systems-I*, *Journal of Electrical and Computer Engineering* and *VLSI Design*, and served in many international conference advisory and technical program committees. He is a Fellow of the IET.



Li Zhang (S'11) received the B.Eng. (Hons) degree in Electrical and Electronics Engineering from Nanyang Technological University (NTU), Singapore, in 2010. He is currently pursuing the Ph.D degree in the division of Circuits and Systems, School of EEE, NTU, Singapore.

His research interests are in hardware security, including hardware IP watermarking and fingerprinting, active IC metering, and hardware Trojan detection and prevention.