Fingerprinting Techniques for Field-Programmable Gate Array Intellectual Property Protection

John Lach, *Member, IEEE*, William H. Mangione-Smith, *Member, IEEE*, and Miodrag Potkonjak, *Member, IEEE*

***Abstract*—As current computer-aided design (CAD) tool and very large scale integration technology capabilities create a new market of reusable digital designs, the economic viability of this new core-based design paradigm is pending on the development of techniques for intellectual property protection. This work presents the first technique that leverages the unique characteristics of field-programmable gate arrays (FPGAs) to protect commercial investment in intellectual property through fingerprinting. A hidden encrypted mark is embedded into the physical layout of a digital circuit when it is placed and routed onto the FPGA. This mark uniquely identifies both the circuit origin and original circuit recipient, yet is difficult to detect and/or remove, even via recipient collusion. While this approach imposes additional constraints on the backend CAD tools for circuit place and route, experiments indicate that the performance and area impacts are minimal.**

***Index Terms*—Field programmable gate array (FPGA), intellectual property protection (IPP), partitioning, physical design, watermarking.**

I. Introduction

WE introduce a fingerprinting technique that applies cryptographically encoded marks to field-programmable gate array (FPGA) digital designs supporting identification of the design origin and the original recipient (i.e., customer of record). The approach is shown to be capable of encoding long messages and to be secure against malicious collusion, while requiring low hardware area and circuit performance overhead.

A. Motivation

It is generally agreed that the most significant problem facing digital integrated-circuit (IC) designers today is system complexity. The process of large system implementation has followed an evolutionary path from multiple ICs, through single ICs, and now into portions of ICs. For example, a 32-bit processor 20 years ago would require several ICs, ten years ago a single IC was necessary, and today a 32-bit reduced instruction-set computer core requires approximately 25% of the StrongARM 110 device developed by Digital Semiconductor in collaboration with ARM Limited [13], [23], [33]. To cope with this increased complexity and to take advantage of localized data and control flows, systems are partitioned into modules, each with a defined system subfunction. This trend toward partitioning has been made more efficient by design reuse, which is essential to reducing development cost, design time, and tapeout risk.

A number of design houses have capitalized on this trend by developing a wide variety of modules, such as parameterized memory systems, input-output channels, arithmetic logic units, and complete processor cores. These reusable modules are collectively known as intellectual property (IP), as they represent the commercial investment of the originating company but do not have a concrete physical manifestation. Consequently, direct theft has become a concern of IP vendors. It may be possible for customers or a third party to sell an IP block as their own without even reverse engineering the design. Because IP blocks are designed to be modular and integrated with other system components, the thief need not understand the architecture or implementation, only the interface and function.

This paper presents a method for deterring such direct theft and misappropriation. The core idea involves embedding in an IP block a digital mark uniquely identifying the design origin and recipient. This mark (origin signature + recipient fingerprint) allows the IP owner to not only verify the physical layout as their property, but to identify the source of misappropriation. This technique provides proof that is more compelling than the existing option of verifying a design against a registered database. This capability is achieved with very low overhead and effort and is secure against multiparty collusion.

Any effective fingerprinting scheme should achieve the following goals.

1) The mark must be difficult to remove.
2) It must be difficult to add a mark to a released design.
3) The mark must be transparent.
4) The mark must have low area and timing overhead and require little added design effort.

The benefits of Properties 1 and 2 can be achieved by integrating the mark into the design. This integration makes the mark more difficult to detect and remove, as there is no clear distinction between the mark and the functional portions of the design. Similarly, this integration makes it more difficult to add postrelease marks. Any attempts to remove the mark or add another incur a great, nearly certain risk of changing the design functionality.

Property 3 is important to provide intellectual property protection (IPP) across a wide community of developers and is the key to extending the benefits of IPP to those who do not employ fingerprinting. By masking the presence of a mark, we discourage all forms of theft. Ayres and Levitt [2] compared the

Manuscript received August 25, 1999; revised April 7, 2001. This work was supported by the Air Force Research Laboratory under Contract F30602-96-C-0350 and Subcontract QS5200 from Sanders, a Lockheed Martin company. This paper was recommended by Associate Editor R. Gupta.

J. Lach is with the Department of Electrical and Computer Engineering, University of Virginia, Charlottesville, VA 22904 USA.

W. H. Mangione-Smith is with the Department of Electrical Engineering, University of California, Los Angeles, CA 90095 USA.

M. Potkonjak is with the Department of Computer Science, University of California, Los Angeles, CA 90095 USA.

Publisher Item Identifier S 0278-0070(01)08880-7.

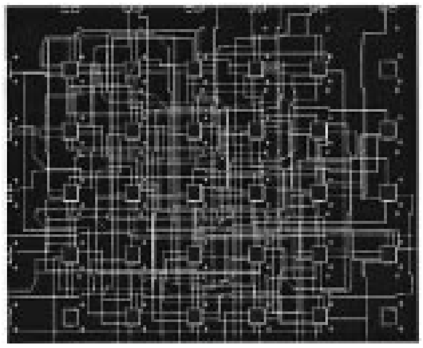
Fig. 1. Original layout.

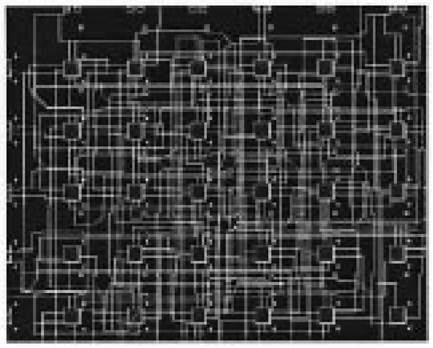
Fig. 2. Marked layout with 96-bit mark.

impact of obtrusive and unobtrusive theft-preventive measures for automobiles. They provide compelling evidence that unobtrusive tracking measures are significantly more effective at reducing theft than measures that are apparent to the thief because of the deterring impact of uncertainty. A parallel concept can be drawn between transparent protection methods such as fingerprinting and more apparent techniques such as conventional design encryption.

Property 4 requires that the overhead in terms of area, timing, and design effort needed to mark the design is not excessive. Efficient IPP must involve a practical and cost effective implementation technique.

B. Motivational Example

Our fingerprinting approach builds upon two existing techniques: an FPGA watermarking technique that hides a mark [20] and an FPGA design tiling and partitioning technique that greatly reduces the cost of generating many different functionally equivalent circuit instances [19]. While the concepts developed here can be applied to a wide range of FPGA architectures, all of the discussion and experimental work are in the context of the Xilinx XC4000 architecture [34]. XC4000 configurable logic blocks (CLBs) each contain two flip-flops and two 16×1 lookup tables (LUTs). A hierarchical and segmented network is used to connect CLBs in order to form a specific circuit configuration.

A secure and transparent mark can be placed in an FPGA design using the previously developed FPGA watermarking technique. Consider the case of PREP Benchmark #4 [26], a state machine that can be mapped into a block of 27 CLBs. Assuming a 6×5 CLB area, this mapping results in three unused CLBs or $3*32 = 96$ unused LUT bits. Each unused LUT bit can be used to encode one bit of the mark. Fig. 1 shows the layout of the original design as produced by the Xilinx backend tools. Fig. 2 shows the layout for the same design after applying the mark constraints to three CLBs and replacing and routing the design over the constraints. The marked CLBs are incorporated into the design, further hiding the mark. Inputs are taken from passing signals in adjacent routing channels and outputs are routed to neighboring *don't care* inputs. Therefore, upon inspection, it is not apparent which CLBs are marked.

This watermarking technique achieves identification of the design source. Tiling [19] is then used to efficiently support fingerprinting (identifying the instance recipient). Consider the Boolean function $Y = (A \vee B) \vee (C \wedge D)$, which might be implemented in a tile containing four logic blocks as shown in Fig. 3(a) This configuration contains one spare logic block and its LUT can hold a mark indicating the owner's signature and recipient's fingerprint. Each recipient could receive this original configuration with a unique fingerprint. Using the same base configuration for a different recipient and, therefore, a different fingerprint would facilitate simple comparison collusion (e.g., XOR) as the only difference between the designs would be the fingerprint. Note, however, that each implementation in Fig. 3(a)–(d) is interchangeable with the original, as the interface between the tile and the surrounding areas of the design is fixed and the function remains unchanged. The timing of the circuit may vary, however, due to the placement and routing changes. With several different instances of the same design, comparison collusion would highlight physical functional differences, thus, disguising the various recipients' fingerprints.

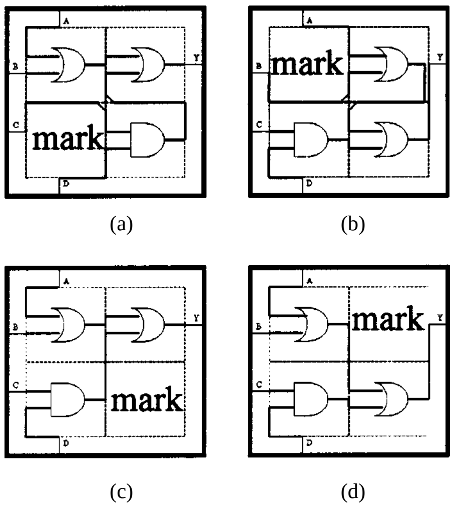

Fig. 3. Four instances of the same function with fixed interfaces.

If a design had four tiles the size of the instance in Fig. 3 (i.e., the design is a 2×2 array of tiles; each tile is a 2×2 array of logic blocks), the total number of logic blocks in the design would be 16. Assuming each logic block has a total of 32 LUT bits and each tile has one logic block free (four unused logic blocks total), 128 LUT bits would be available to encode a mark. Each tile has four instances (four possible locations for the unused block), making the total number of design instances (configurations of unused blocks) $4^4 = 256$. A nontiled design would have $\binom{16}{4} = 1820$ possible instances (configurations of unused blocks), but each instance requires a complete execution of the backend computer-aided design (CAD) software. It can be assumed that placing and routing the entire design requires X amount of design effort and that each tile instance only requires approximately $\mathrm{X}/4$ amount of effort. However, each tile instance can be used in $4^3 = 64$ different design instances when combined with the other tile instances. Therefore, the effective effort required to generate each tile instance is $\mathrm{X}/(4^*64)$ and each instance of the total tiled design requires $\mathrm{X}/64$ amount of effort.

C. Paper Organization

The remainder of the paper is organized as the following. Section II discusses work related to this fingerprinting approach and Section III details the approach itself. Section IV evaluates the approach through experimentation and the paper is summarized in Section V.

II. Related Work

The bulk of existing IPP is composed of encrypted sources files. Hardware description language (HDL) models are encrypted so that they can be loaded into authorized simulators or synthesis tools without making the source forms visible to the system designer who uses the IP block. In theory, this technique allows IP vendors to provide soft macros and high performance simulation models without ever exposing their property to the risk of theft; the CAD tools maintain the design integrity. In practice, most of these approaches have eventually been broken, often times by attacking the CAD tool directly. While stronger forms of encryption and more thorough systems engineering will likely improve the reliability of encryption approaches, there is no compelling reason to expect a reliable approach to become available in the near future.

Recently, work has been proposed for the protection of digital circuit IP through watermarking at various levels in the design hierarchy using the superimposition of additional constraints in conjunction with those specified by the user. The work presented here is the first physical design watermarking and fingerprinting technique for FPGA designs. The majority of IP is already programmable components and all economic and historical trends indicate that the importance of these components will continue to rise.

Ad-hoc techniques for marking text and image documents have been manually practiced for many centuries [4]. Recently, techniques for hiding signature data in image, video, and audio signals have received a great deal of attention. A spectrum of steganographic techniques for protection of digital images has been proposed [8], [11], [28]. While many of the initial techniques for marking images were not able to provide proper proof of ownership [12], several recent techniques provide protection against all known attacks [30].

Although protection of digital audio signals emerged as a more difficult task, at least three different techniques have been proposed [3], [7], [11]. The protection of video streams using steganography has been demonstrated by several European and US research teams [15], [16], [29].

Recently, a set of techniques has been proposed for the protection of digital circuit IP through watermarking at various levels in the design hierarchy using the superimposition of additional constraints in conjunction to those specified by the user [9], [10], [17], [18], [24], [31]. In this paper, we propose the first fingerprinting technique for FPGA designs at the physical design level. Addressing the design at a lower level of abstraction has one additional advantage: all designs are significantly larger at lower levels of abstraction, enabling more and/or larger marks that are more difficult to detect and remove.

There have been a number of fingerprinting efforts reported in data hiding and cryptographic literature for various forms of IP (e.g., audio, video, and other forms of digital data) [5], [6], [25]. These techniques greatly enhance the protection of both buyers and merchants of digital artifacts. All of these techniques are targeting protection of still artifacts, such as image and audio streams. This paper is novel in that it addresses IPP using fingerprinting for ICs.

III. Approach

Watermarking inserts digital signatures in unused logic block LUTs as introduced in the motivational example in Section I-B. Results indicate that the area and timing overhead required for inserting large marks in a design is low and that the marks are secure against detection or removal without reverse engineering [20].

Directly applying the watermarking technique to fingerprinting (i.e., replacing the design origin signature with the recipient's fingerprint for each copy) is susceptible to recipient collusion. Performing a simple comparison between the two bitstreams would reveal the fingerprints as the only difference, which when removed would yield a fully functional yet unmarked circuit.

This vulnerability can be avoided by taking advantage of the flexible nature of FPGAs to create physical differences among functional components of the designs. By moving the location of the fingerprint for each instance of the design (i.e., reserve different logic blocks for the fingerprint), the functional components will also have a different layout (e.g., as shown in Fig. 3). Therefore, all comparisons that are done yield physical functional differences and any attempt to remove the differences would yield a useless circuit. However, generating an entire layout for each instance of the design would require replacing and routing the entire circuit. Tiling requires that only a small portion of the design be changed, greatly reducing the amount of backend CAD tool effort for each design instance.

A. Watermark Preparation, Embedding, and Validation

The process of mark embedding and extraction is detailed in the flowchart shown in Fig. 4. Marks are embedded at design time for each design instance recipient. The mark extraction process is used if a design owner suspects a theft or misappropriation.

The first step of signature preparation and embedding [see Fig. 4(a)] is encoding the authorship signature and recipient

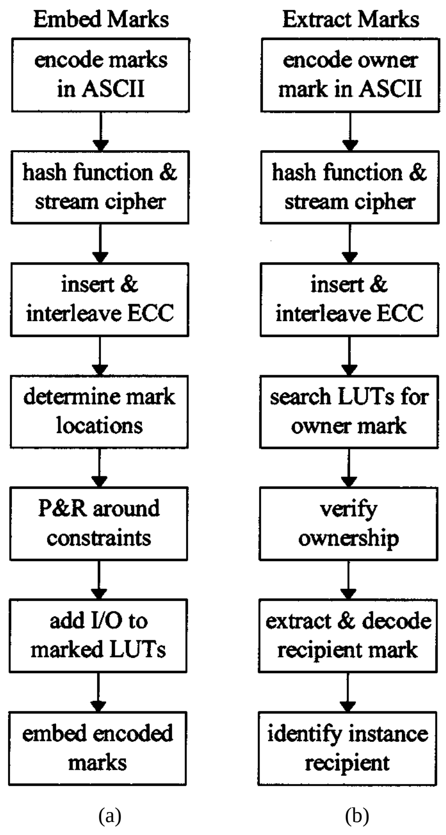

Fig. 4. (a) Embed flow. (b) Extract flow.

fingerprint (denoted collectively as "marks") as 7-bit ASCII strings. The strings input to the fingerprinting system for embedding in the circuit and are later produced by the verification program. The strings are first compressed using a cryptographic hash function and a stream cipher is applied. (Specifically, we use for signature preparation the cryptographic hash function MD5 and the stream cipher RC4 [22], [27] on which many of today's state-of-the-art cryptographic commercial programs are based.) As a consequence of the cryptographic functions, the mark is whitened so as not to look like any particular statistical spectrum. This whitening of the marks does not mask their content, but rather their existence. With a flat distribution, the marks are more difficult to detect.

The next step in signature preparation involves adding error-correction coding (ECC). By introducing ECC into the marks, we combat the malicious third party that manages to identify a part of a mark and attempts to modify or remove it. A fundamental tradeoff exists between the number of bits allocated to the mark and ECC and the sum must not exceed an estimate of the bits available for encoding. We use a fault-tolerant encoding scheme that is similar to that used for marking file system structures on disk drives [21]—the system decides upon a level of ECC protection based on the available free space and records these settings. Using this approach, each design has an appropriate amount of ECC while still guaranteeing that a generic verification system will be able to retrieve the proper signature.

The final step in signature preparation involves interleaving multiple ECC blocks. Consider the case where all mark information is encoded in the 16×1 LUTs of a Xilinx XC4000 device. It is possible that a malicious third party would be able to identify a particular LUT that is nonessential to the device function and change its programming. If 16 consecutive ECC blocks are interleaved one bit at a time over a set of LUTs, then each LUT will only contain one bit from any ECC. This interleaving guarantees the validation software can successfully retrieve the signature in the face of any single point fault, i.e., a tampered LUT.

Mark locations are determined by tile instances. Each instance is generated by placing the unused logic blocks in a unique location. The CAD tool then places and routes the rest of the tile around the constraints. Therefore, the mark locations in each total design instance are unique. However, the owner signature and recipient fingerprint are always placed in a constant location relative to one another. For example, one LUT of a logic block can be used to embed the owner signature and the other to embed the fingerprint. This simplifies the verification process.

After the mark locations are determined, the LUTs to be implanted with the marks are given arbitrary inputs and outputs in order to further disguise the marks. The inputs are taken from signals in adjacent interconnect channels and the outputs route to neighboring logic block *don't care* inputs. The selection of adjacent signals for inputs and *don't cares* for outputs is layout dependent, as the added signals should be kept short. This incorporation helps to hide the marked LUTs without severely impacting design performance, as the dummy outputs are not a functional design component. The performance impact it does have due to interconnect capacitance and fan-out is measured by a timing analyzer. These results are detailed in Section IV. When design instances are compiled for release, the marks are embedded in the predefined LUTs by reprogramming the respective bits in the bit stream. Therefore, the final step of mark embedding is an entirely postprocessing step.

The first essential element in mark validation is retrieving the FPGA configuration. This step is straightforward for FPGAs based on static memory, as they are loaded over an external bus that can be monitored. It may not always be possible to retrieve configurations for technologies based on antifuses or flash memories. For this reason, we have focused our technique on static memory devices.

Because the source of a suspected misappropriation is unknown, the mark locations are unknown. The IP vendor must provide the ownership string (denoted in the graph as "mark") that they claim to be embedded in the design and the signature preparation process is applied to determine the corresponding embedded mark, so the independent validation team can search for it. The verification team then searches the LUTs for the embedded mark. If all or a large percentage of the mark is found, ownership has been established. Once the locations of the ownership marks are known, the instance recipient fingerprint can be extracted and applied to the reverse signature preparation process, as the recipient fingerprint is in a constant location relative to the ownership signature. The relevant LUT bits are decrypted using the known key and printed out, establishing the source of theft or misappropriation.

1) Limitations: The standard digital design flow follows these steps: behavioral HDL, synthesis to register transfer language, technology mapping, and finally physical layout involving place and route. Marks can be applied to any level in this design flow and, if developed properly, will propagate to later stages. However, because a mark is nonfunctional, it may be removed by reverse engineering a design to a level higher than where the mark was applied. Fortunately, most FPGA vendors will not reveal the specification of their bit stream configurations (i.e., the correlation of the bit stream to the physical resources), specifically to complicate the task of reverse engineering and, thus, protect the investment of their customers [32]. In addition, FPGA manufactures claim that the reverse engineering process for their devices is difficult [14]. Most devices follow a form of Pareto's rule: the first 80% of the configuration information can be determined relatively easily by inspection, the next 16% is much more difficult and the final details are extremely difficult to establish. The complexity is enhanced by an irregular pattern that is not consistent between rows or columns, as a result of the hierarchical interconnect network.

While the task of reverse engineering an FPGA design is difficult, it is possible to crack the configuration specification with a concerted effort. NeoCAD Inc. was able to accomplish this for the Xilinx XC4000 series devices through a directed investigation of the output produced by the Xilinx backend tools. Another line of attack involves removing the packaging material and successive layers and using image processing inspection to produce a circuit representation of the logic blocks. This approach has been used to produce a complete layout of a 386 microprocessor in approximately two weeks [1].

As a consequence of the proven success of reverse engineering, we believe that hiding the mark is necessary, but not sufficient. Any effective fingerprinting scheme should make the mark appear to be part of the functional digital circuit (as described in Section I-B) to whatever extent possible.

B. Fingerprinting

Tiling divides a design into a set of tiles that possess the same characteristics as the example in Fig. 3, i.e., each tile has a specific functionality and a locked interface to the rest of the design (i.e., the neighboring tiles). Therefore, any signals crossing tile boundaries are restricted to remain on the same piece of interconnect. Several instances of each tile are generated and each instance can replace another one without affecting the rest of the circuit (except timing) due to the locked interface. The various tile instances are then matched to create one instance of the entire design. This reduces the total number of instances needed to be generated and vastly reduces the effort and memory required to produce and store each instance.

After each instance for each tile is generated, the instances are prepared for marking and evaluated for timing statistics. Depending on the timing specifications of the design, some instances may be discarded from the pool of potential tile instances. The remaining instances are collected in a database. For example, MCNC benchmark c499 can be divided into six tiles each with eight instances, creating the possibility for $8^6 = 262\,144$ different instances of the total design. When a copy of the design is needed, an instance for each tile is extracted from the database and the marks are inserted in the unused logic blocks with the postprocessing step described above.

A colluding group may be able to find that they have instance matches among some of their tiles, thus, allowing for tile comparison collusion. However, it is extremely unlikely that matches will be found among all (or even a large portion) of the tiles. Furthermore, the tile structure and boundaries are not inherent properties of the FPGA configuration and are, therefore, not generally apparent to the colluders. The colluding recipients may be able to remove a portion of their fingerprints, but the majority will remain intact. The key to this approach is efficiently introducing variation among the functional parts of the designs, so that collusion cannot be used to separate common functional components from unique fingerprints.

The following pseudocode summarizes the approach.

```
1.   create initial nonfingerprinted design;
2.   extract timing and area information;
3.   while (!complete) {
4.      partition design into tiles;
5.      if (!(mark size && collusion
          protection)) break;
6.      for (i=1; i<=# of tiles; i++) {
7.         for (j=1; j <= # of tile
             instances; j++) {
8.            create tile instance (i,j);
9.            if (instance meets timing
                criteria) {
10.              incorporate unused logic
                   blocks into design;
11.              store instance;
12.   } } } }
13.   for (i=1;i<= # of recipients;i++) {
14.      prepare mark(i);
15.      select tile instances from database;
16.      insert mark in unused LUTs;
17.   }
```

Lines 1 and 2 initialize the process by establishing the physical layout for the nonfingerprinted design on which all area and timing overhead is based. Lines 3–12 perform the tiling technique, creating a database of tile instances. The variables for this section are mark size, collusion protection (level of security based on presumed number of collaborating attackers), and timing requirements. Mark size and collusion protection affect the tiling approach, while the timing requirements define the instance yield. Individual tile instances are accepted contingent upon their meeting the timing requirements or discarded from the pool of potential tile instances. Lines 13–17 are executed for each distributed instance of the design. Line 14 derives the unique recipient fingerprint with asymmetric fingerprinting techniques that require the cooperation of both the designer and the instance recipient, ensuring security for both [5], [25].

C. Tiling Optimization

The tiling technique performed in lines 3–12 involves selecting a tile size (x) that best balances the security of the fingerprint and the design effort required for the fingerprinting process. The selection of x is based on a set of constants and user performance requirements, including the desired emphasis on either security or development effort.

The constants are the total design area (d) and the logic block utilization ($a = \#$ unused logic blocks/# total logic blocks). The variables defined by the user are the timing specifications (which impact instance yield (y), the percentage of tile instances that meet the timing specifications), a collusion group size (n) and the maximum percentage of the mark that can be removed while leaving enough of the fingerprint intact for unique recipient identification (b). These five criteria lead to preliminary calculations: the number of tiles in the design ($t = d/x$), the average number of free logic blocks per tile ($f = \lfloor x^*a \rfloor \approx x^*a$), the number of instances per tile ($i = \binom{x}{f}$), the number of instances per tile that meet the timing constraints ($j = i^*y$), and the approximate design effort required for the fingerprinting technique ($e = t^*x^*i = d^*i$). Design effort refers to the number of complete passes through the CAD tools (i.e., configurations of the entire design) that are required. Fig. 5 helps to show the relation of the variables.

The above information can be used to calculate the odds that n people could collude to remove their fingerprints in one tile

$$c_1 = 1 - \frac{(j)!}{((j-n)!)^*(j)^n}. \tag{1}$$

Equation (1) assumes that the only way to remove a fingerprint from a tile is to find two matching tile instances, thus making the recipient fingerprint the only difference between two tiles (i.e., reverse engineering is not considered here). A simple XOR comparison between the two tiles would remove the two fingerprints, thus creating an instance which can be distributed to all participating colluders. The owner's signature remains in all tiles, however.

The process of removing the fingerprint from one tile can be repeated until a certain portion of the complete design has been cleared of recipient fingerprints. A binomial calculation indicates the odds that $b\%$ of the fingerprint can be removed

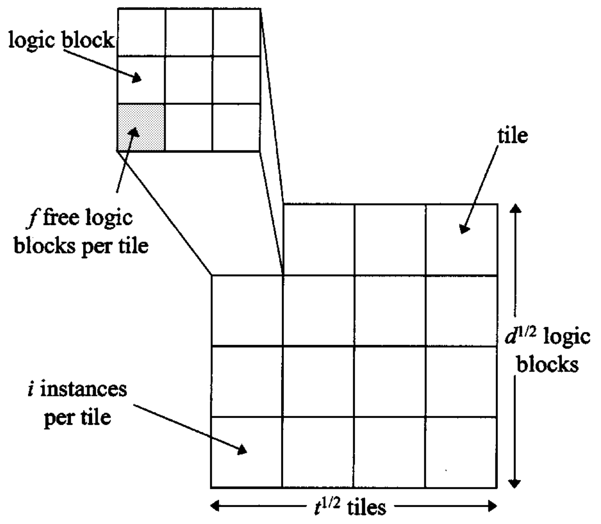

Fig. 5. Partitioned physical layout with corresponding variables.

$$c_{\%} = \sum_{k=b^*t}^{t} \binom{t}{k} {}^*(c_1)^k {}^*(1-c_1)^{t-k}. \tag{2}$$

The security of the fingerprint improves with a larger tile size (x) as a result of the greater number of instances

$$\binom{d}{f^*t} > \binom{x}{f}^* t.$$

However, as mentioned above, larger tile sizes require more effort (e) to generate. The product of security and design effort yields (3), shown at the bottom of the page.

This equation reveals that there is a critical value for x that balances the tradeoff between security and design effort. Certain applications may put a stronger emphasis on one or the other. For today's FPGA applications, design effort is one of the prime limiting factors. FPGA place and route is extremely time consuming, thus forcing an emphasis on limiting design effort.

$$\begin{aligned} c_{\%}{}^*e = & d^*\binom{x}{x^*a} \\ & * \sum_{k=b^*d/x}^{d/x} \left[\binom{\frac{d}{x}}{k} \right. \\ & * \left(1 - \frac{\left[\binom{x}{x^*a}{}^*y\right]!}{\left[\binom{x}{x^*a}{}^*y - n\right]!^* \left[\binom{x}{x^*a}{}^*y\right]^n} \right)^k \\ & \left. * \left(\frac{\left[\binom{x}{x^*a}{}^*y\right]!}{\left[\binom{x}{x^*a}{}^*y - n\right]!^* \left[\binom{x}{x^*a}{}^*y\right]^n} \right)^{(d/x)-k} \right] \end{aligned} \tag{3}$$

TABLE I
VARIATION OF RESOURCES USED AMONG TILE INSTANCES

Design	Original # of CLBs	Final # of CLBs	Final – Original / Original
9sym	46	49	.065
c499	94	96	.021
c880	110	115	.045
duke2	93	100	.075
rd84	27	28	.037
planet1	95	100	.053
styr	78	81	.038
s9234	195	206	.056
sand	82	90	.098

Also, today's applications see configuration bitstreams being distributed directly to approved recipients, thus requiring any collusion effort to bring n number of people together. Any one person would have a difficult time collecting a larger number of distributed instances of the design.

Conversely, trends in FPGA technology and applications may put a reduced emphasis on design effort and a greater emphasis on security. FPGA CAD tools continue to improve and workstations on which the software runs are improving tremendously. Therefore, in the future, design effort may not be as onerous. Also, one foreseeable application of FPGA designs includes distributing designs over the internet, either directly to recipients or available as a form of hardware application. In either situation, one person could gain access to a great many instances of the design, either by eavesdropping on a network line or downloading numerous instances of the application, each time receiving a different fingerprint. Then, n becomes not the number of people colluding, but the number of instances that are being compared, possibly by one person. This possibility raises the importance of fingerprint security.

IV. Experimental Results

To evaluate the area and timing overhead of the approach, we conducted an experiment on nine MCNC designs implemented on the Xilinx XC4000 architecture. For design effort and fingerprint security, the following constants were assumed: design area $(d) = 400$ Xilinx XC4000 CLBs, number of unused CLBs/number of total CLBs $(a) = 0.1$, and instance yield $(y) = 0.9$.

The overhead of the proposed approach comes in the form of area (physical resources), timing, and design effort. Area overhead is inevitable, as previously unused LUTs are used to encode the mark. Table I shows the area of the designs before and after the application of the fingerprinting approach. (Note: the device size used was 400 CLBs, but constraints were applied restricting the place-and-route areas to a size close to that of the mapped design.) A number of factors complicate the task of calculating the physical resource overhead. The place-and-route tools indicate the number of CLBs that are used for a particular placement. However, these utilized CLBs rarely are packed into a minimal area. Unused CLBs introduce flexibility into the place-and-route step that may be essential for completion or good performance. For example, the initial c880 design possesses a concave region that contains 42 utilized CLBs, but also

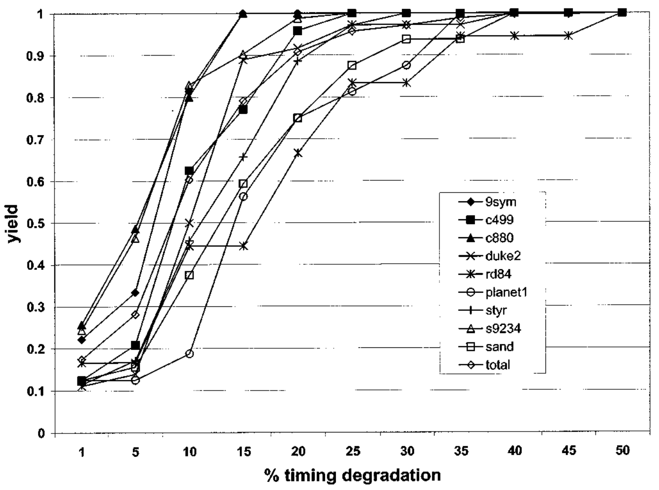

Fig. 6. Instance yield versus timing specifications.

ten unutilized CLBs (19%). Therefore, we will report overhead in terms of the area used by the fingerprinted design minus the total area of the original design, including unused CLBs such as the 19% measured above. The average, median and worst case area overheads were 5.4%, 5.3%, and 9.8%, respectively. The size of the mark (owner signature + recipient fingerprint) that can be encoded is dependent on this overhead (32 bits per CLB). If a larger mark is desired and more area overhead is acceptable, additional CLBs can be used.

Timing overhead may arise due to the constraints on physical component placement as defined by the size and location of the mark. An LUT dedicated to the mark may impede placement of circuit components and lengthen the critical path. As the mark size grows relative to the design size, more constraints are made on the placement of the design, thus increasing the possibility for performance degradation.

Timing overhead is shown in Fig. 6. For each design, the instance yield (i.e., number of tile instances that meet the timing specifications/total number of tile instances) is shown as the timing specifications (measured as percent increase over the original, nonfingerprinted design timing) grow more lenient. The results reveal that a 20% increase in timing yields approximately 90% of total tile instances as acceptable. Preserving a smaller user defined timing degradation results in a lower instance yield and vice versa. Relatively small changes in a circuit netlist or routing constraints can often result in a dramatically different placement and a corresponding change in speed. It appears that the impact of fingerprinting on performance is comparable to this characteristic variance.

As the equations in Section III-C indicate, effort (e) increases with tile size (x). Design effort may often be the limiting factor in tile size selection, despite the fact that fingerprint security increases with tile size. By examining the chance that an increasing number of colluders have to remove their recipient fingerprints from one tile (1) and all tiles [(2) with $b = 100\%$] by direct comparison collusion with varied tile sizes, it is clear that larger tile sizes drastically reduce the chance that any portion of the fingerprint may be removed. However, even for a small tile size of 40 and a large group of 200 people colluding, there is only one chance in approximately five million that the entire

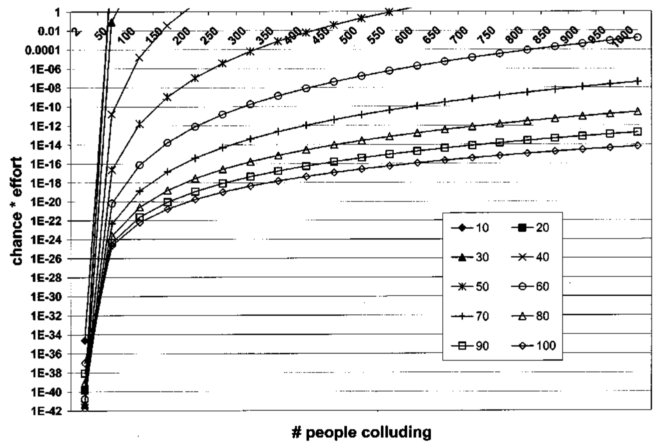

Fig. 7. Chance to break all tiles * design effort.

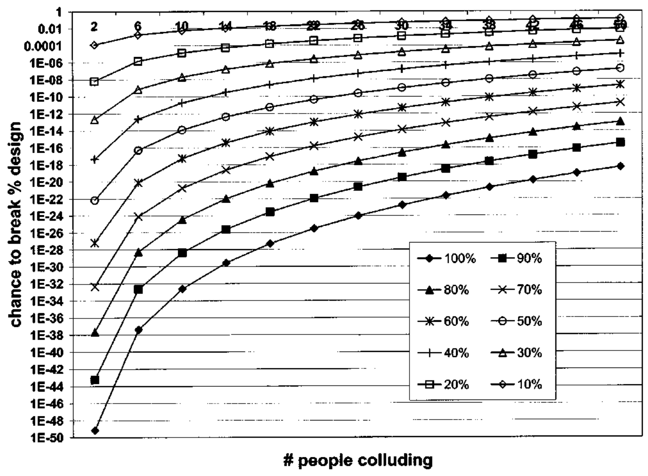

Fig. 9. Chance to break variable % of fingerprint (tile size $= 40$).

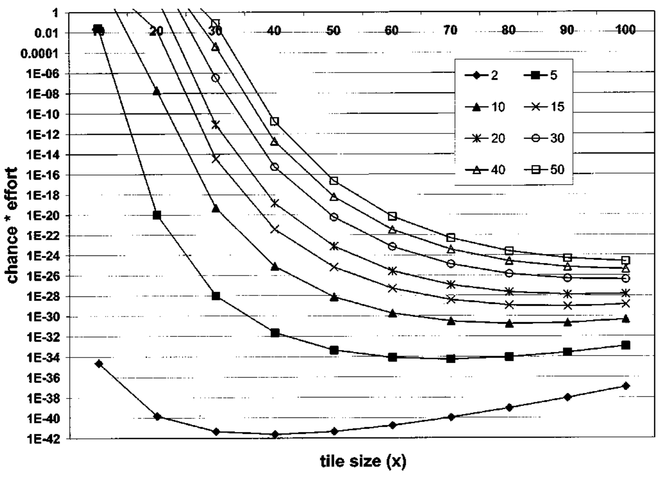

Fig. 8. Tile-size critical value.

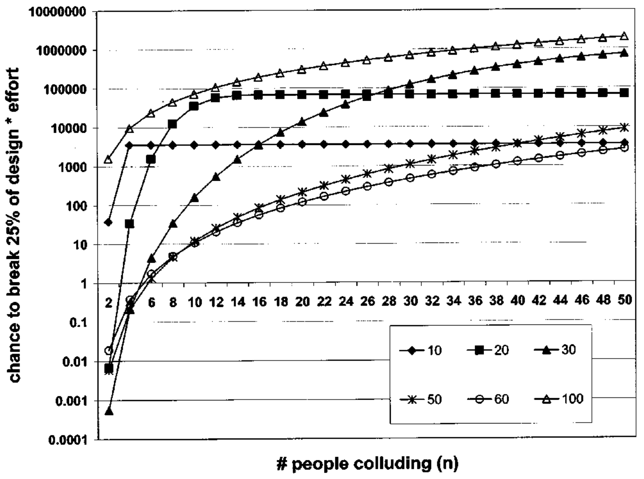

Fig. 10. Chance to break 25% of fingerprint * effort.

fingerprint could be removed. It, therefore, is important to compare the tradeoff between design effort and fingerprint security in a proper manner. Fig. 7 is a direct multiplication comparison (3) given $d = 400$, $a = 0.1$, and $y = 0.9$.

Other comparisons may be more relevant based on the application and design setting. As mentioned above, current applications do not require as much security, as a large number of design instances are not easily available to a group of colluders. Also, current design settings place a strong emphasis on design effort, as modern FPGA place-and-route technology is time consuming. Therefore, a different comparison (e.g., security*effort^2) may present a more useful measure of current needs. Future applications may require a different comparison, perhaps placing a greater emphasis on fingerprint security and a smaller emphasis on effort. Fig. 7, however, reveals that for a large number of colluders and direct multiplication comparison, it is better to have a larger tile size.

Fig. 8 shows the same data as Fig. 7, but it instead plots various collusion sizes against tile size and focuses in on a smaller, more reasonable number of colluders. The minimum value for each plot is the critical value denoting the optimal tile size for the particular collusion group size. The best tile size actually is a midsized tile for a small collusion group (e.g., $n = 2 \Rightarrow$ best x; $n = 10 \Rightarrow$ best $x = 80$) and the specific optimum tile size increases for larger collusion groups.

Fig. 9 displays the chance that a growing number of colluders have to remove a certain percentage of the fingerprint for a tile size (x) of 40. Even for a small tile size such as 40, it remains extremely unlikely that a colluding group could remove even a small portion of the recipient fingerprint. The chance that 15 colluders would be able to remove 30% of the recipient fingerprint by comparison collusion is one chance in approximately four million.

Fig. 10 directly multiplies security by design effort for each tile size, revealing that the optimal tile size for a growing number of colluders is predominantly midsized tiles because the fingerprint security remains extremely high for midsized tiles while requiring significantly less design effort.

V. Conclusion

As digital IC design complexity increases, forcing an increase in design reuse and third party macro distribution, IPP will become more important. The fingerprinting approach presented here creates such protection for FPGA IP by inserting a unique marker identifying both the origin and recipient of a design. The fingerprinting process produces a secure mark and a unique physical layout for each design instance recipient. However, little extra design effort is required to generate multiple physical designs, as a physical design partitioning technique (tiling) is used, which minimizes the backend CAD tool effort for each physical design iteration. Although the mark is applied to the physical layout of the design by imposing constraints on the backend CAD tools, experiments reveal that the area and timing overhead is low.

REFERENCES

[1] R. Anderson and M. Kuhn, "Tamper resistance—A cautionary note," in *Proc. USENIX Workshop Electronic Commerce*, Nov. 1996, pp. 1–11.

[2] I. Ayres and S. D. Levitt, "Measuring positive externalities from unobservable victim precaution: An empirical analysis of Lojack," Nat. Bureau Economic Res., Cambridge, MA, NBER Working Paper No. W5928, Feb. 1997.

[3] W. Bender *et al.*, "Techniques for data hiding," *IBM Syst. J.*, vol. 35, no. 3-4, pp. 313–336, 1996.

[4] H. Berghel and L. O'Gorman, "Protecting ownership rights through digital watermarking," *IEEE Comput.*, vol. 27, pp. 101–103, Sept. 1996.

[5] I. Biehl and B. Meyer, "Protocols for collusion-secure asymmetric fingerprinting," in *Proc. Symp. Theoretical Aspects of Computer Science*, Mar. 1997, pp. 399–412.

[6] D. Boneh and J. Shaw, "Collusion-secure fingerprinting for digital data," *IEEE Trans. Inform. Theory*, vol. 44, no. 5, pp. 1897–1905, Sept. 1998.

[7] L. Boney *et al.*, "Digital watermarks for audio signals," in *Proc. Int. Conf. Multimedia Computing and Systems*, June 1996, pp. 473–480.

[8] J. Brassil and L. O'Gorman, "Watermarking document images with bounding box expansion," in *Proc. Int. Workshop Information Hiding*, June 1996, pp. 227–235.

[9] A. E. Caldwell *et al.*, "Effective iterative techniques for fingerprinting design IP," in *Proc. Design Automation Conf.*, June 1999, pp. 843–848.

[10] E. Charbon, "Hierarchical watermarking in IC design," in *Proc. Custom Integrated Circuits Conf.*, May 1998, pp. 295–298.

[11] I. J. Cox *et al.*, "Secure spread spectrum watermarking for images, audio and video," in *Proc. Int. Conf. Image Processing*, Sept. 1996, pp. 243–246.

[12] S. Craver *et al.*, "Can invisible watermarks resolve rightful ownership?," *Proc. SPIE-Int. Soc. Opt. Eng.*, vol. 3022, pp. 310–321, Feb. 1997.

[13] S. Furber, *ARM System Architecture*. Reading, MA: Addison-Wesley, 1996, pp. 329–329.

[14] R. Goering, "IP98 forum exposes struggling industry—Undefined business models, unstable core prices cited," *EE Times*, no. 1000, Mar. 1998.

[15] F. Hartung and B. Girod, "Copyright protection in video delivery networks by watermarking of pre-compressed video," in *Proc. Eur. Conf. Multimedia Applications, Service, and Techniques*, May 1997, pp. 423–436.

[16] ——, "Watermarking of MPEGk2 encoded video without decoding and re-encoding," *Proc. SPIE-Int. Soc. Opt. Eng.*, vol. 3020, pp. 264–274, Feb. 1997.

[17] I. Hong and M. Potkonjak, "Behavioral synthesis techniques for intellectual property protection," in *Proc. Design Automation Conf.*, June 1999, pp. 849–854.

[18] A. B. Kahng *et al.*, "Watermarking techniques for intellectual property protection," in *Proc. Design Automation Conf.*, June 1998, pp. 776–781.

[19] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "Low overhead fault-tolerant FPGA systems," *IEEE Trans. VLSI*, vol. 6, pp. 212–221, June 1998.

[20] ——, "Signature hiding techniques for FPGA intellectual property protection," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1998, pp. 186–189.

[21] M. K. McKusick *et al.*, "A fast system for UNIX," *ACM Trans. Comput. Syst.*, vol. 2, no. 3, pp. 181–197, 1984.

[22] A. J. Menezes, P. C. V. Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL: CRC, 1996.

[23] J. Montanaro *et al.*, "A 160 MHz 32b 0.5 W CMOS RISC Microprocessor," in *Proc. Int. Solid-State Circuits Conf.*, Feb. 1996, pp. 49–62.

[24] A. Oliveira, "Robust techniques for watermarking sequential circuit designs," in *Proc. Design Automation Conf.*, June 1999, pp. 837–842.

[25] B. Pfitzmann and M. Waidner, "Anonymous fingerprinting," in *Proc. Int. Conf. Theory and Application of Cryptographic Techniques*, May 1997, pp. 88–102.

[26] "PREP Benchmark Suite #1, Version 1.3," Programmable Electronic Performance Group, Los Altos, CA, 1994.

[27] B. Schneier, *Applied Cryptography: Protocols, Algorithms and Source Code in C*. New York: Wiley, 1996.

[28] J. Smith and B. Comiskey, "Modulation and information hiding in images," in *Proc. Int. Workshop Information Hiding*, June 1996, pp. 207–226.

[29] G. A. Spanos and T. B. Maples, "Performance study of a selective encryption scheme for the security of networked, real-time video," in *Proc. Int. Conf. Computer Communications and Networks*, Sept. 1995, pp. 2–10.

[30] A. H. Tewfik and M. Swanson, "Data hiding for multimedia personalization, interaction and protection," *IEEE Signal Processing Mag.*, vol. 14, pp. 41–44, July 1997.

[31] I. Torunoglu and E. Charbon, "Watermarking-Based copyright protection of sequential functions," in *Proc. Custom Integrated Circuits Conf.*, May 1999, pp. 35–38.

[32] S. Trimberger, private communication, 1997.

[33] J. Turley, "ARM Grabs Embedded Speed Lead," *Microprocessor Rep.*, vol. 10, Feb. 1996.

[34] Xilinx, Inc., *The Programmable Logic Data Book*. San Jose, CA: Xilinx, 1996.

John Lach (S'99–M'00) received the B.S. degree in science, technology, and society from Stanford University, Stanford, CA, in 1996 and the M.S. and Ph.D. degrees in electrical engineering from the University of California, Los Angles (UCLA), in 1998 and 2000, respectively.

Since 2000, he has been an Assistant Professor in the Electrical and Computer Engineering Department, University of Virginia, Charlottesville. He is the Principal Investigator on the dynaptable computing project funded by the National Science Foundation. His current research interests include real-time embedded systems, computer-aided design techniques for very large scale integration, general-purpose and application-specific processor design, intellectual property protection, and field programmable gate arrays.

Dr. Lach received the 2001-2002 University of Virginia Teaching Fellowship and two School of Engineering and Applied Sciences Dean's Awards from UCLA.

William H. Mangione-Smith (M'95) received the B.S.E. degree in electrical engineering and the M.S.E. and Ph.D. degrees in computer science and engineering from the University of Michigan, Ann Arbor, in 1987, 1992, and 1992, respectively.

From 1991 to 1995, he was with Motorola, Inc., where he participated in the design of the Envoy Personal Digital Assistant. In 1995, he joined the Electrical Engineering Department at the University of California, Los Angeles, where he is currently an Associate Professor. He is a Co-Princiapal Investigator on the Mojave configurable computing program and served as the Program Chair for MICRO 26. His current research interests include using dynamic circuits to implement configurable computing systems, low-power processor and system design, multimedia and communications processing, and all techniques for leveraging instruction-level parallelism.

Dr. Mangione-Smith is a member of the ACM. He received the National Science Foundation CAREER Award in 1998.

Miodrag Potkonjak (S'90–M'91) received the Ph.D. degree in electrical engineering and computer science from University of California, Berkeley, in 1991.

In 1991, he joined the Computer and Communicatons Research Laboratories, NEC USA, Princeton, NJ. He joined the University of California, Los Angeles (UCLA), in 1995, where he is currently a Professor with the Computer Science Department. He has served on a number of program committees, including the International Conference on Image Processing, the Design Automation Conference, and the International Conference on Computer-Aided Design. He has authored or coauthored approximately 200 papers in journals and conferences and holds five patents. His recent watermarking-based intellectual property protection (IPP) research formed a basis for the Virtual Socket Initiative Alliance (IPP) standard. His current research interests include embedded systems, sensor networks, computational security, and intellectual property protection.

Dr. Potkonjak received the National Science Foundation CAREER Award, the Okawa Foundation Award, UCLA TRW SEAS Excellence in Teaching Award, and a number of best paper awards.