Fingerprinting Digital Circuits on Programmable Hardware

John Lach[1], William H. Mangione-Smith[1], Miodrag Potkonjak[2]
University of California, Los Angeles

Department of Electrical Engineering[1]
56-125B Engineering IV
Los Angeles, CA 90095
{jlach, billms}@icsl.ucla.edu

Department of Computer Science[2]
4532K Boelter Hall
Los Angeles, CA 90095
{miodrag}@cs.ucla.edu

Abstract. Advanced CAD tools and high-density VLSI technologies have combined to create a new market for reusable digital designs. The economic viability of the new core-based design paradigm is pending on the development of techniques for intellectual property protection. A design watermark is a permanent identification code that is difficult to detect and remove, is an integral part of the design, and has only nominal impact on performances and cost of design.

Field Programmable Gate Arrays (FPGAs) present a particularly interesting set of problems and opportunities, because of their flexibility. We propose the first technique that leverages the unique characteristics of FPGAs to protect commercial investment in intellectual property through fingerprinting. A hidden encrypted message is embedded into the physical layout of a digital circuit when it is mapped into the FPGA. This message uniquely identifies both the circuit origin and original circuit recipient, yet is difficult to detect and/or remove. While this approach imposes additional constraints on the back-end CAD tools for circuit place and route, experiments involving a number of industrial-strength designs indicate that the performance impact is minimal.

1 Introduction

We introduce a fingerprinting technique that applies cryptographically encoded marks to Field Programmable Gate Array (FPGA) digital designs in order to support 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. Nonetheless, the technique is efficient and requires low overhead in terms of hardware area and circuit performance.

1.1 Motivation

It is generally agreed that the most significant problem facing digital 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, twenty years ago a 32-bit processor would require several ICs, ten years ago a single IC was necessary, and today a 32-bit RISC core requires approximately 25% of the StrongARM 110 device developed by Digital Semiconductor in collaboration with ARM Limited [1-3]. Fortunately, complex

systems tend to be assembled using smaller components in order to reduce complexity as well as to take advantage of localized data and control flows. This trend toward partitioning enables design reuse, which is essential to reducing development cost and risk while also shortening design time. While systems designers have employed design reuse for years, what is new is that the boundaries for component partitions have moved inside of the IC packages.

A number of design houses have appeared that provide a wide range of modules, such as parameterized memory systems, I/O channels, ALUs 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 natural physical manifestation.

Direct theft is 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 does not need to understand either the architecture or implementation.

This paper presents a deterrent to such direct misappropriation. The essential idea involves embedding a digital mark, which uniquely identifies the design origin and recipient, in an IP block. 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, in a way that is likely to be much more compelling than the existing option of verifying the 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 after releasing the IP to a customer.
3. The mark should be transparent.
4. The mark should have low area and timing overhead and little design effort.

The benefits of properties 1 and 2 are readily apparent and can be achieved by integrating the mark into the design. It then becomes more difficult to detect and remove, as there is no clear distinction between the mark and parts necessary to the design, thus making it more difficult to add another mark. Any attempts to remove the mark or add another incur a much greater risk of changing the design function.

Property 3 is important to provide IP protection across a wide community of developers and is the key to extending the benefits of IP protection to those who do not employ fingerprinting. By masking the presence of a mark, we discourage all forms of theft. Ayres and Levitt compared the impact of obtrusive and unobtrusive theft-preventive measures for automobiles [4]. 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. We believe that a parallel exists between measures 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.

1.2 Motivational Example

Our fingerprinting approach builds upon two existing techniques: an FPGA watermarking technique that hides a mark [5] and an FPGA design tiling and partitioning technique that greatly reduces the cost of generating many different circuit instances which are functionally equivalent [6].

FPGAs are programmable logic devices that are composed of an array of configurable logic blocks (CLBs) which are connected via a programmable network. A device is configured with a bitstream generated by CAD tools specifying the functionality of the CLBs and the routing of the network. While the concepts developed here can be applied to a wide range of FPGA architectures, all of the discussion and experimental work will be conducted in the context of the Xilinx XC4000 architecture [7]. CLBs in an XC4000 each contain two flip-flops and two 16x1 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 a previously developed FPGA watermarking technique [5]. Consider the case of PREP Benchmark #4 [8], a large state machine, which can be mapped into a block of 27 CLBs. This mapping results in 3 unused CLBs, or 3*32=96 unused LUT bits. Each unused LUT bit is used to encode one bit of the mark. Figure 1 shows the layout of the original design as produced by the standard Xilinx backend tools, while Figure 2 shows the layout for the same design after applying the mark constraints to the three unused CLBs and re-mapping the design. The marked CLBs are incorporated into the design with unused network connections and neighboring CLB inputs, further hiding the mark.

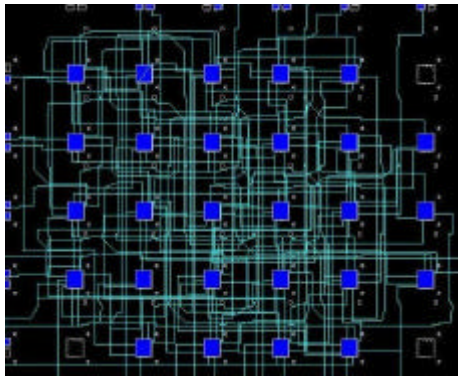

Figure 1. Original layout of PREP benchmark #4

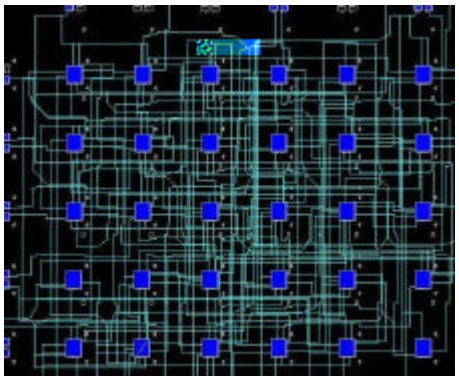

Figure 2. Marked layout of PREP benchmark #4 with a 96-bit mark

Tiling [6] is then used to efficiently support fingerprinting. Consider the Boolean function $Y=(A \wedge B) \wedge (C \vee D)$, which might be implemented in a tile containing four CLBs as shown in Figure 3.I. This configuration contains one spare CLB, 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 Figure 3.I-IV 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 changes in routing. With several different instances of the same design, comparison collusion would highlight functional differences, thus disguising the differences between the various recipients' fingerprints.

If a design had four tiles the size of the instance in Figure 3 (i.e. the design is a 2x2 array of tiles; each tile is a 2x2 array of CLBs), the total number of CLBs in the design would be 16. Assuming each CLB has a total of 32 LUT bits and each tile has one CLB free, 128 LUT bits would be available to encode a mark. Each tile has four instances, making the total number of design instances 4^4=256. A non-tiled design would have $\binom{16}{4}$= 1820 possible instances, but each instance requires a complete execution of the backend CAD software which may require X amount of design effort. Each tile instance only requires X/4 amount of effort, but each instance can be used in 4^3=64 different design instances. Therefore, the effective effort required to generate each tile instance is X/(4*64), and each instance of the total tiled design requires X/64 amount of effort.

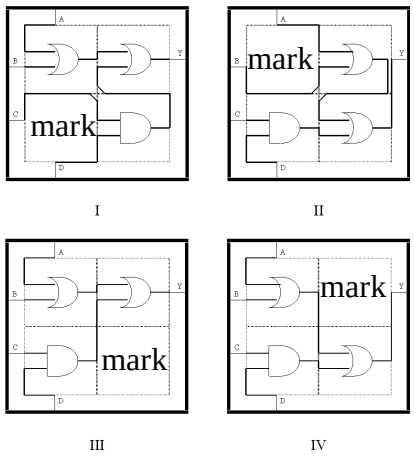

Figure 3. Four instances of the same function with fixed interfaces

1.3 Limitations

The standard digital design flow follows these steps: behavioral hardware description language (HDL), synthesis to register transfer language (RTL), technology mapping, and finally physical layout involving place and route. Marks can be applied to any level of 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 stage in the flow before the mark has been applied. Fortunately, most FPGA vendors will not reveal the specification of their configuration streams, specifically to complicate the task of reverse engineering and thus protect the investment of their customers. For example, the Xilinx XC4000

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, etc. The complexity is enhanced by an irregular pattern that is not consistent between rows or columns, as a result of the hierarchical interconnect network. Xilinx does not take any specific actions to make their configurations difficult to reverse engineer. However, they do believe that it is difficult to do in general, and they promise their customers that they will keep the bitstream specification confidential in order to raise the bar for reverse engineering [9].

1.4 Contributions

This paper presents the first fingerprinting method for protecting reusable digital circuit IP, even after the IP has been delivered in commercial products. By manipulating hardware resources, we are able to encode relatively long messages in a manner that is difficult to observe by a third party, resists tampering and collusion, and has little impact on circuit performance or size. This capability provides three main benefits:

1. It reduces the risk that a circuit will be stolen, i.e. used illegally without payment or transferred to a third party.
2. It identifies not only the origin of the design, but also the origin of the misappropriation.
3. It can be used to identify the backend tool chain used to develop a design, and thus be part of the royalty mechanism used for CAD tools.

1.5 Paper Organization

The remainder of the paper is organized in the following way. Section 2 discusses work related to this fingerprinting approach, and Section 3 details the approach itself. Section 4 evaluates the approach through experimentation, and the work is summarized in Section 5.

2 Related Work

Related work can be divided into three main areas: digital watermarking, fingerprinting, and reverse engineering.

Ad-hoc techniques for marking text and image documents have been manually practiced for many centuries [10]. 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 [11-13]. While many of the initial techniques for marking images were not able to provide proper proof of ownership [14], several recent techniques provide protection against all known attacks [15].

Although protection of digital audio signals emerged as a more difficult task, at least three different techniques have been proposed [12, 16, 17]. The protection of video streams using steganography has been demonstrated by several European and US research teams [18-20].

Recently, a set of techniques has been proposed for protection of digital circuit IP through watermarking at the behavioral level using superimposition of additional constraints in conjunction to those specified by the user [21]. In this paper, we

propose the first fingerprinting technique for FPGA designs. The majority of intellectual property is already programmable components, and all economic and historical trends indicate that the importance of these components will continue to rise. Finally, note that addressing the design at a lower level of abstraction has one additional advantage. All designs are significantly larger at lower levels of abstraction, enabling marks that are more difficult to detect and remove.

There has been a number of fingerprinting efforts reported in data hiding and cryptographic literature [22-24]. A spectrum of protocols has been established which 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. To the best of our knowledge, this work is the first effort that addresses IP protection using fingerprinting.

While Xilinx and other FPGA vendors make some efforts to complicate the task of reverse engineering, it certainly 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. Given this information, it should be relatively straightforward to produce a Xilinx netlist file and then use commercial tools to move back up the design flow. Another line of attack involves removing the packaging material and successive layers and using image processing inspection to produce a circuit representation of the CLB. This approach has been used to produce a complete layout of a 386 microprocessor in approximately 2 weeks [25].

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 to whatever extent is possible.

3 Approach

The watermarking technique is the general approach of inserting marks in unused CLB LUTs as introduced in the motivational example above. Results [5] indicate that the area and timing overhead required for inserting large marks in a design is low and that the approach is secure against most attacks.

Directly applying the watermarking technique to fingerprinting (i.e. replace the design origin signature with the recipient's fingerprint for each copy) is susceptible to collusion. Performing a simple comparison between the two bitstreams would reveal that the only differences were due to the individual fingerprints. Removing the differences would yield a fully functional yet unmarked circuit.

This vulnerability can be avoided by taking advantage of the flexible nature of FPGAs to create differences among functional components of the designs. By moving the location of the fingerprint for each instance of the design (i.e. reserve different CLBs for the fingerprint), the functional components will also have a different layout. Therefore, all comparisons that are done yield 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 a trip through the place-and-route tools for the entire circuit. Tiling requires that only a small portion of the design be changed. The algorithm divides a design into a set of tiles that possess the same characteristics as the example in Figure 3.

That is, each tile has specific functionality and a locked interface to the rest of the design. Several instances of each tile can be generated, and each instance can replace another without affecting the rest of the circuit (except timing) due to the locked interface. The various tile instances can then be matched to create one instance of the entire design. This reduces the total number of instances that need to be generated, and vastly reduces the effort and memory required to produce each instance.

The design tiling algorithm was originally developed in an eye toward fault-tolerance, but with the same goal of effort and memory reduction [6]. For fault-tolerance, different instances of each tile reserve different CLBs as unused. In the face of a CLB fault, the appropriate instance can be activated without affecting the rest of the circuit. The same result could be achieved by storing a great many instances of the entire design, leaving various CLBs free in each instance, but the effort to place and route each instance and the memory required to store each instance makes this approach impractical.

Much in the same way that tiling for fault-tolerance reduces the effort required to generate the various fault-tolerant instances and the memory to store them, tiling also makes fingerprinting more efficient and practical.

3.1 Watermark Preparation, Embedding, and Validation

We use cryptography tools to generate a set of FPGA physical designs (configurations) which correspond to the signature of the author of the design. The application of cryptographic techniques ensures also cryptographically strong hiding and low correlation of the added features.

The first step of the signature preparation and embedding process is encoding of the authorship signature as a 7-bit ASCII string. The signature string is given to the fingerprinting system for embedding in the circuit and is later produced by the verification program. The string is first compressed using a cryptographic hash function. The output is processed using public-key encoding. Finally, to produce a message that corresponds to the initial signature, we use a stream cipher. Note, in such a way, we generate a signature of arbitrary length.

Specifically, we use for the signature preparation the cryptographic hash function MD5, the public-key cryptosystem RSA, and the stream cipher RC4 [26, 27], on which many of today's state-of-the-art cryptographic commercial programs are based.

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

The final step in signature preparation involves interleaving multiple ECC blocks. Consider the case where all signature information is encoded in the 16x1 LUTs of a Xilinx XC4000 device. It is possible that a malicious third party would be able to

identify a particular LUT that is non-essential to the device function and change its programming. If sixteen 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 that the validation software can successfully retrieve the signature in the face of any single point fault, i.e. a LUT that has been tampered with.

Validation involves retrieving the signature embedded in the configuration. The essential element to validating any signature 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 be possible to retrieve the configuration for technologies based on anti-fuse or flash-memory, though the issue remains open given recent successful efforts at reverse engineering. For this reason, we have focused our technique on static memory devices.

When the owners of an IP block believes their property has been misappropriated, they must deliver the configuration in question to an unbiased validation team. The IP vendor produces a seed that they claim was used to produce the block. With the seed and signature, the validation team reverses the signature preparation and embedding process: identify the CLBs used for hiding the signature based on the specific tile instances of the suspected instance of the complete design, reverse the block interleaving, apply the ECC if necessary, decrypt the message using a known key, and finally print out the resulting signature. If the signature matches that claimed by the IP vendor, then the source of the misappropriation has been established.

Essentially, the owners must demonstrate that encoded constraints match their encrypted signature. Therefore, they must provide the validation team with their original and encrypted signatures. The encrypted signature can be verified to match the constraints in the realized design, but this procedure assumes that the third party (validation team) will not reveal the signature to others later. We plan to address zero-knowledge proofs for signatures (where this restriction is removed) in future efforts.

3.2 Fingerprinting

After each instance for each tile is generated, the instances are prepared for marking. Every unused CLB is incorporated into the design with unused network connections and neighboring CLB inputs, and timing statistics are generated for each instance. Depending on the timing specifications of the design, some instances may be discarded. The remaining instances are collected in a database. For example, MCNC benchmark c499 can be divided into 6 tiles, each with 8 instances, creating the possibility for $8^6 = 262{,}144$ different instances of the total design.

When a copy of the design in needed, an instance for each tile is extracted from the database and the recipient's fingerprint is inserted in the unused CLBs.

A group of people colluding may be able to find that they have instance matches among some of their tiles, thus allowing for tile comparison collusion, but it 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 generally apparent to the colluders, as they are not an inherent property of the FPGA configuration. Therefore, the colluding recipients may be able to remove a portion of their fingerprints, but the majority of the fingerprints will remain intact. The key to this approach is efficiently introducing wide variation among the functional parts of the

designs, so that collusion cannot be used to separate common functional components from unique fingerprints.

The pseudo-code in Figure 4 summarizes the approach.

```
1.    create initial non-fingerprinted 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 CLBs 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.   }
```

Figure 4. Pseudo-code for fingerprinting approach

Lines 1 and 2 initialize the process by establishing the physical layout for the non-fingerprinted 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 collaborators), and timing requirements. Mark size and collusion protection affect the tiling approach, while the timing requirements define the instance yield (i.e. individual tile instances are accepted contingent upon their meeting the timing requirements). Lines 13-17 are executed for each distributed instance of the design. Line 14 derives the unique recipient fingerprint with asymmetric fingerprinting techniques [23, 24].

3.3 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 given constants and user defined variables, including the desired emphasis on either security or effort.

The constants are the FPGA device size (d) and the CLB utilization (a = # unused CLBs/# total CLBs). The variables defined by the user are the timing specifications (which impact instance yield (y), the percent of tile instances that meet the timing specifications), a collusion set 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 number of free CLBs 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 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.

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

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

Equation 1. Odds to remove fingerprint from one tile

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. 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 cleaned. A binomial calculation indicates the odds that b% of the fingerprint can be removed:

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

Equation 2. Chance to remove fingerprint from b% of the tiles

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:

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

Equation 3. Design effort times odds to remove fingerprint from b% of the tiles

This equation reveals that there is a critical value for x which 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 mapping is extremely time consuming, thus forcing an emphasis on limiting design effort. 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 won't be as onerous. Also, one forseable 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.

4 Experimental Results

To evaluate the area and timing overhead of the approach, we conducted an experiment on nine MCNC designs. For design effort and fingerprint security, the following constants were assumed: device size (d) = 400 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 1 shows the area of the designs before and after the application of the fingerprinting approach. A number of factors complicate the task of calculating the physical resource overhead. The place-and-route tools will 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 10 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% measure above. The average, median and worst-case area overheads were 5.4%, 5.3%, and 9.8% respectively. The size of the mark (signature + fingerprint) that can be encoded is dependent on this overhead. If a larger mark is desired, extra CLBs can be added thus increasing overhead.

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

Table 1. Variation of resources used among instances for each tile

Timing overhead may arise due to the constraints on physical component placement as defined by the size and location of the mark. A 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 show in Figure 5. 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, non-fingerprinted design timing) grow more lenient. The results reveal that a 20% increase in timing yields approximately 90% of total tile instances as acceptable. 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 below this characteristic variance.

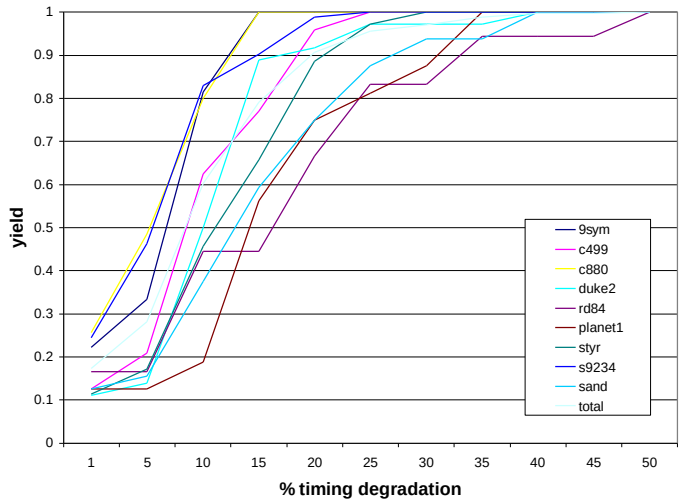

Figure 5 Instance yield vs. timing specifications

As the equations in Section 3.3 indicate, effort (*e*) increases with tile size (*x*). Design effort may often be the limiting factor in tile size selection, despite the fact

that security increases with tile size. By examining the chance that an increasing number of colluders have to break one tile (Equation 1) and all tiles (Equation 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. Even for a tile size of 40 and 200 people colluding, there is only one chance in approximately 5 million that the entire fingerprint could be removed. It therefore is important to compare the tradeoff between design effort and fingerprint security in a proper manner. Figure 6 is a direct multiplication comparison (Equation 3). 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 aren't easily available to a group of colluders. Also, current design settings place a strong emphasis on design effort, as modern FPGA mapping technology is time consuming. Therefore, a different comparison (e.g. security * effort^2) may present a more useful measure of current needs. As mentioned above, future applications may require a different comparison, perhaps placing a greater emphasis on fingerprint security and a smaller emphasis on effort. Figure 6, however, reveals that for a large number of colluders and direct multiplication comparison, it is better to have a larger tile size.

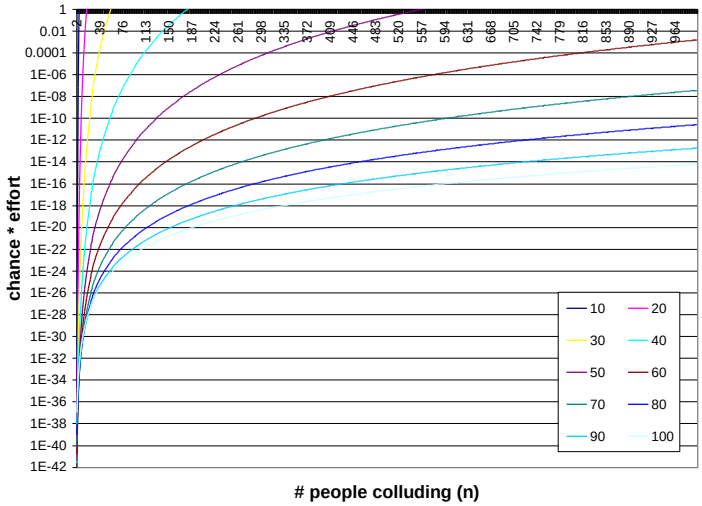

Figure 6. Chance to break all tiles * design effort

Figure 7 shows the same data as Figure 6, 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. It is easy to see here, that the best tile size actually is a mid-sized tile for a small collusion group (e.g. n=2 => best x=40; n=10 => best x=80), and the specific optimum tile size increases for larger collusion groups.

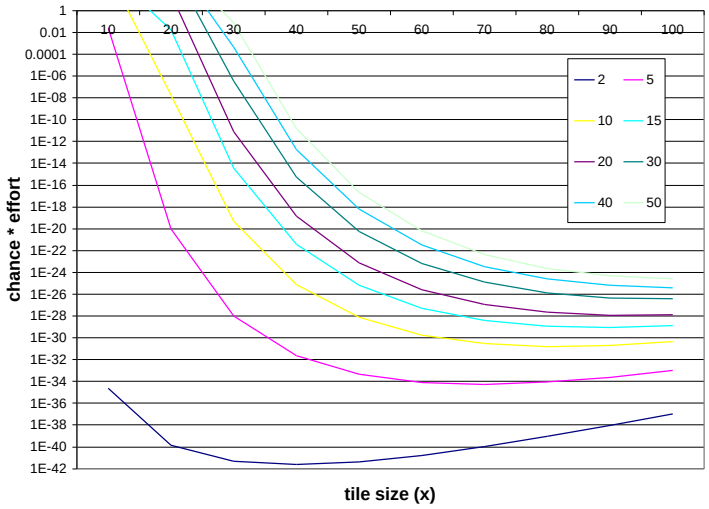

Figure 7. Tile size critical value

Figure 8 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 fingerprint. The chance that 15 colluders would be able to remove 30% of the fingerprint by comparison collusion is one chance in approximately 4 million.

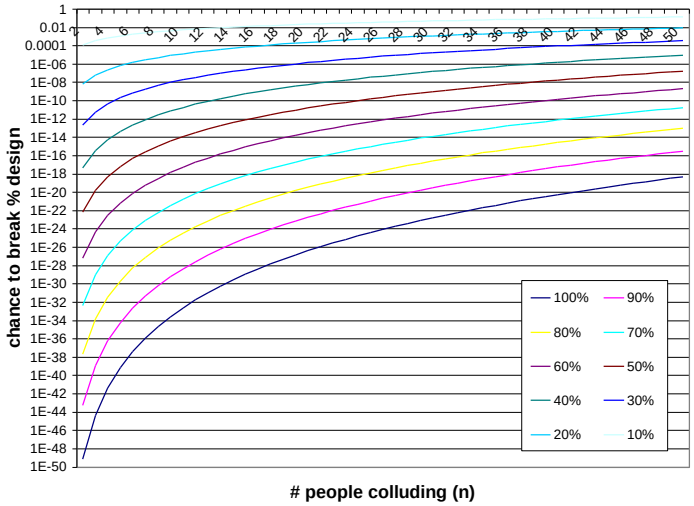

Figure 8. Chance to break % of fingerprint (tile size = 40)

Figure 9 directly multiplies security by design effort for each tile size revealing that the optimal tile size for a growing number of colluders is predominantly mid-sized tiles because the fingerprint security remains extremely high for mid-sized tiles while requiring significantly less design effort.

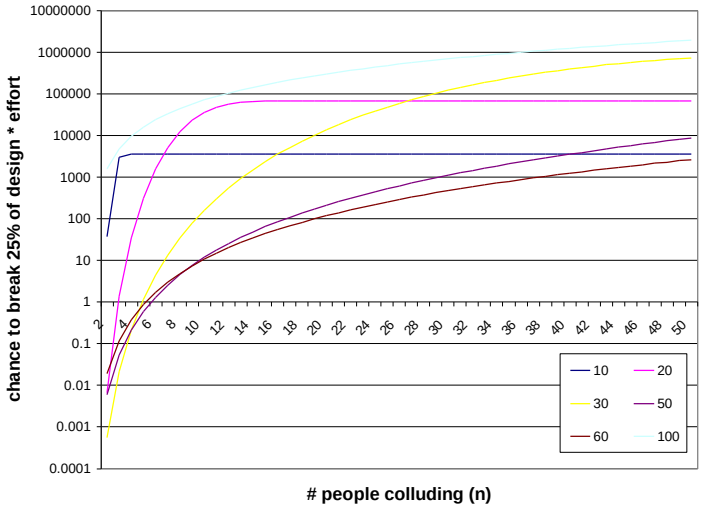

Figure 9. Chance to break 25% of fingerprint * effort

5 Conclusion

As digital IC design complexity increases, forcing an increase in design reuse and third party macro distribution, intellectual property protection will become more important. The fingerprinting approach presented here creates such protection for FPGA intellectual property by inserting a unique marker identifying both the origin and recipient of a design. The fingerprinting process produces an extremely secure mark (chance of removing a fingerprint is always less than one in a million) but requires little extra design effort. 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 extremely low.

Acknowledgements

The authors would like to thank Gang Qu for his assistance. This work was supported by the Defense Advanced Research Projects Agency of the United States of America, under contract DAB763-95-C-0102 and subcontract QS5200 from Sanders, a Lockheed Martin company.

References

[1] J. Turley, "ARM Grabs Embedded Speed Lead," *Microprocessor Report*, vol. 10, 1996.

[2] J. Montanaro et al., "A 160MHz 32b 0.5W CMOS RISC Microprocessor," *Proc. of International Solid-State Circuits Conference*, 1996.

[3] S. Furber, *ARM System Architecture*, Menlo Park: Addison-Wesley, 1996, p. 329.

[4] I. Ayres and S. D. Levitt, "Measuring Positive Externalities from Unobservable Victim Precaution: An Empirical Analysis of Lojack," *The Economics Review*, 1997.

[5] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "Signature Hiding Techniques for FPGA Intellectual Property Protection," submitted to *ICCAD '98*, 1998.
[6] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "Low Overhead Fault-Tolerant FPGA Systems," *IEEE Transactions on VLSI*, vol. 6, 1998.
[7] Xilinx, *The Programmable Logic Data Book*, San Jose, CA, 1996.
[8] Programmable Electronic Performance Group, "PREP Benchmark Suite #1, Version 1.3," Los Altos, CA, 1994.
[9] S. Trimberger, Personal Communication, Xilinx Corporation, 1997.
[10] H. Berghel and L. O'Gorman, "Protecting Ownership Rights Through Digital Watermarking," *IEEE Computer*, 1996, pp. 101-103.
[11] J. Brassil and L. O'Gorman, "Watermarking Document Images with Bounding Box Expansion," *First International Workshop on Information Hiding*, Cambridge, U.K., 1996.
[12] I. J. Cox et al., "Secure Spread Spectrum Watermarking for Images, Audio and Video," *International Conference on Image Processing*, 1996.
[13] J. Smith and B. Comiskey, "Modulation and Information Hiding in Images," *First International Workshop on Information Hiding*, Cambridge, U.K., 1996.
[14] S. Craver et al., "Can Invisible Watermarks Resolve Rightful Ownership?" *The International Society for Optical Engineering*, 1997.
[15] A. H. Tewfik and M. Swanson, "Data Hiding for Multimedia Personalization, Interaction, and Protection," *IEEE Signal Processing Magazine*, 1997, pp. 41-44.
[16] W. Bender et al., "Techniques for Data Hiding," *IBM Systems Journal*, vol. 35, 1996, pp. 313-336.
[17] L. Boney et al., "Digital Watermarks for Audio Signals," *International Conference on Multimedia Computing and Systems*, 1996.
[18] G. A. Spanos and T. B. Maples, "Performance Study of a Selective Encryption Scheme for the Security of Networked, Real-Time Video," *International Conference on Computer Communications and Networks*, 1995.
[19] F. Hartung and B. Girod, "Copyright Protection in Video Delivery Networks by Watermarking of Pre-Compressed Video," *ECMAST '97*, 1997.
[20] F. Hartung and F. Girod, "Watermarking of MPEG-2 Encoded Video Without Decoding and Re-Encoding," *Multimedia Computing and Networking*, 1997.
[21] I. Hong and M. Potkonjak, "Behavioral Synthesis Techniques for Intellectual Property Protection," unpublished manuscript, 1997.
[22] D. Boneh and J. Shaw, "Collusion-Secure Fingerprinting for Digital Data," *CRYPTO '95*, 1995.
[23] I. Biehl and B. Meyer, "Protocols for Collusion-Secure Asymmetric Fingerprinting," *STACS '97, 14th Annual Symposium on Theoretical Aspects of Computer Science*, 1997.
[24] B. Pfitzmann and M. Waidner, "Anonymous Fingerprinting," *International Conference on the Theory and Application of Cryptographic Techniques*, 1997.
[25] R. Anderson and M. Kuhn, "Tamper Resistance - A Cautionary Note," *USENIX Electronic Commerce Workshop*, 1996.
[26] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. New York: John Wiley & Sons, 1996.
[27] A. J. Menezes, P. C. V. Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.