Robust FPGA Intellectual Property Protection Through Multiple Small Watermarks

Tracks M2.1 and M4.3

John Lach (presenter)
56-125B Engineering IV
University of California
Los Angeles, CA 90095
Phone: (310) 794-1630
Fax: (310) 825-7928
jlach@icsl.ucla.edu
UCLA Electrical Engineering Department

William H. Mangione-Smith (contact)
56-125B Engineering IV
University of California
Los Angeles, CA 90095
Phone: (310) 206-4195
Fax: (310) 825-7928
billms@icsl.ucla.edu
UCLA Electrical Engineering Department

Miodrag Potkonjak
3532G Boelter Hall
University of California
Los Angeles, CA 90095
Phone: (310) 825-0790
Fax: (310) 794-5056
miodrag@cs.ucla.edu
UCLA Computer Science Department

All appropriate organizational approvals for the publication of this paper have been obtained. If accepted, the authors will prepare the final manuscript in time for inclusion in the Conference Proceedings and will present the paper at the Conference.

John Lach

Robust FPGA Intellectual Property Protection Through Multiple Small Watermarks

Abstract – A number of researchers have proposed using digital marks to provide ownership (watermarking) and recipient (fingerprinting) identification for intellectual property. Many of these techniques share three specific weaknesses: complexity of copy detection, vulnerability to mark removal after revelation for ownership verification, and mark integrity issues due to partial mark removal. This paper presents a method for watermarking and fingerprinting field programmable gate array (FPGA) intellectual property (IP) that achieves robustness by responding to these three weaknesses. The key techniques involve using secure hash functions to generate and embed multiple small marks that are more detectable, verifiable, and secure than existing IP protection techniques.

1 Introduction

1.1 Motivation

Design reuse has grown due to the continual increase in digital IC system complexity. While twenty years ago a 32-bit processor would require several ICs, a 32-bit RISC core currently requires approximately 25% of the StrongARM 110 device developed by Digital Semiconductor in collaboration with ARM Limited [7,18,24]. Design partitioning allows complex systems to be assembled from smaller modules. Although this type of design reuse has been employed for years, the boundaries of these modules have recently moved inside IC packages. Reused modules include parameterized memory systems, I/O channels, ALUs, and complete processor cores.

Design reuse has led to the rise of Intellectual Property Protection (IPP) concerns [8]. IP modules are often designed by one company and sold in a non-physical form (e.g. HDL, netlist, layout) to others, and therefore do not have a natural physical manifestation. The IP blocks are modular and are designed to be integrated within other systems, usually on the same chip. As a result of the flexible, intangible nature of these modules, IP theft has become a problem. A thief need only resell or reuse an IP module without even reverse engineering the design, as proof of IP ownership is difficult to assert due to its inherently abstract nature.

Existing FPGA design watermarking [16] and fingerprinting [14] techniques attempt to deter such direct theft and misappropriation of FPGA IP. Digital marks are embedded in a design layout which identify the design origin (watermarking and fingerprinting) and specific design instance recipient (fingerprinting only). These existing FPGA IP protection techniques focused more on embedding the mark and keeping it secure. Little emphasis was placed on the mark itself and the quality and complexity of its detection and verification.

Our new techniques enable efficient copy detection for identifying stolen or misappropriated IP, even if it is embedded in a complex system. The techniques also provide simple mark extraction and convincing design source and instance recipient verification, without threatening the security of other designs. Finally, the embedded marks are more secure against removal attacks and more robust against partial mark removal. The new techniques provide this greater efficiency, verifiability, security, and robustness for protecting FPGA IP without increasing the user design effort, CAD tool effort, or area and timing overhead required by the existing FPGA watermarking and fingerprinting techniques.

1.2 Existing FPGA IPP Techniques

FPGA design watermarking embeds a digital mark in unused lookup tables (LUTs) throughout the design [16]. These LUTs are incorporated into the design with unused interconnect and neighboring logic block

"don't care" inputs, further hiding the signature. Watermarks have been shown to be secure against removal unless the design can be reverse engineered to the netlist level, thus protecting against direct IP theft. Fingerprinting combines watermarking and a physical design partitioning technique [15]. Using solely the watermarking technique to embed recipient information would facilitate simple comparison collusion[1]. Therefore, fingerprinting efficiently generates a different design configuration for each recipient using the physical design partitioning technique. Each recipient receives a different, yet functionally equivalent, set of partitioned blocks. Any attempt to eliminate differences in non-identical blocks via comparison collusion would render the design non-functional.

1.3 Copy Detection

Scanning a large set of diverse FPGA designs for potentially stolen or misappropriated copies can be a laborious process if comparisons are based on functional similarities or widely dispersed watermarks. Fortunately, existing FPGA IPP techniques restrict mark placement to logic block LUTs, and it can be quickly established where LUTs are located in an FPGA bitstream. Therefore, LUTs can be scanned for ownership marks, making copy detection reasonably efficient.

The new FPGA IPP techniques allow for even more efficient copy detection than existing techniques provide by allowing each design and every copy of each design to have the same set of watermarks. Therefore, when a set of designs is being searched, a constant, constrained set of marks is being compared to the LUT configurations, greatly reducing the search complexity. Mark security is maintained by the use of multiple watermarks (existing FPGA IPP techniques use a single large watermark) as discussed in Sections 1.4 and 1.5.

In addition, searching for multiple marks does not increase design search complexity by a significant measure even if a single mark were used in every design. This efficiency is achieved by using small marks, specifically marks that fit in the target architecture's LUTs. Therefore, for each LUT examination, the set of marks is iteratively compared to the LUT contents instead of successive LUT-sized portions of the large single mark. For example, given 16-bit LUTs, comparing 20 successive 16-bit portions of a single 320-bit mark to LUT contents requires the same complexity as comparing 20 16-bit marks.

The use of many small marks makes copy detection significantly more efficient.

1.4 Mark Verification

Publicly verifying design ownership has previously presented a risk to the design owners. Once a mark was publicly revealed for verification, it increased the possibility that other design recipients could find and remove the mark. This vulnerability is especially crucial if the same mark is used for all of the owners' IP including unrelated modules, which is necessary for efficient copy detection.

The new techniques eliminate this concern with the use of multiple watermarks. A subset of the marks is revealed for public verification, providing enough information for proof of ownership yet not enough for other recipients to remove a significant amount of marks in their copies. For example, an IP module may be developed by one company and sold to 100 others. The module designers could embed a set of small watermarks in each design. If a theft or misappropriation is suspected, a subset of the watermarks could be publicly provided by the designer for ownership verification without risking that the other 99 companies (as well as the customers for all the other designs created by the company that possess the same set of marks) receive enough information to remove all, or even a large portion, of the design ownership marks. The subset may be removed, but the others will remain.

In addition, an important characteristic of any watermarking approach involves the owner being able to precisely locate the marks for ownership verification based on a predefined extraction technique. If the

[1] Two recipients could simply XOR their bitstreams, as the only difference between the designs would be their unique fingerprints within.

owner tells an independent verification team that a watermark is in locations F(seed), for some known algorithm F() and an integer seed value, they will have more credibility than if they tell the team to search the entire design until they find a watermark. The specific approach to such an implementation for watermarking verification is detailed in Section 4.4.

Mark location is not precisely known for fingerprinting, for which there is variability among designs released to multiple partners for comparison collusion prevention. Therefore, mark location is a function of both a seed value and the design recipient. Assuming the design recipient is not known before verification, it is necessary to make an exhaustive comparison between the ownership watermark and all possible mark locations (i.e. all LUTs), much in the same way that copy detection searches are performed. Ownership can still be established in this method, albeit less convincingly. However, searching possible locations for fingerprints would not be efficient, as the set of recipients could be large. Without knowing which customer distributed the design, we would have to consider each of them against the contents of each LUT. We address this point by co-locating ownership watermarks with recipient fingerprints. The most direct method is (given multiple LUTs per logic block) to use one LUT in a logic block to store the ownership watermark and the other to store the fingerprint. Therefore, once the ownership mark location is established via the exhaustive LUT search, recipient marks can be extracted and verified with the same efficiency and level of validity as the normal F(seed) watermark verification technique.

1.5 Mark Security and Robustness

Our new techniques embed multiple small marks, as opposed to a single large mark, creating greater mark security and robustness than existing techniques provide.

Smaller marks reduce the possibility that a mark may be partially removed, thus increasing mark robustness for verification. For example, if ten of twenty LUTs containing the mark were erased by reverse engineering, one 320-bit mark with 50% of its bits removed is less compelling for ownership verification than ten intact 16-bit marks.

Fingerprinting robustness is also enhanced by the use of smaller marks. If the physical partition boundaries are established by a collusion group, partition blocks may be exchanged. This may break up large fingerprints, rendering them unreadable. Similarly, fingerprints in otherwise identical partition blocks may be removed by comparison collusion (see Section 5.4 for security against comparison collusion information), thus destroying parts of large signatures. Shorter signatures contained entirely in a LUT maintain their integrity, as each exchanged tile passes all of the design owner and recipient information. Similarly, the short signatures in non-identical partition blocks remain intact after the others are removed by comparison collusion.

Multiple distinct marks also increase mark security by reducing the possibility of repetition-based statistical attacks from the repetition of a single small watermark. For example, a single 16-bit mark repeated twenty times is susceptible to such an attack, while ten to twenty distinct marks would be less apparent.

1.6 Contributions

We present new FPGA design watermarking and fingerprinting techniques for IPP that provide for more efficient copy detection, more convincing ownership and recipient verification, and more secure and robust marks without increasing user design effort, CAD tool effort, or area and timing overhead.

1.7 Paper Organization

Sections 2 and 3 provide the technical background and related work information necessary for understanding these new IPP techniques. Section 4 details the new approaches to FPGA IPP, and Section 5 evaluates the new approaches. Section 6 concludes the discussion with summarizing comments.

2 Technological Issues

2.1 Vulnerability to Reverse Engineering

The concern over mark removal through reverse engineering for older methods still exists for the new techniques. Marks can be applied to any level in the design flow, including behavioral hardware description language (HDL), synthesis to register transfer language (RTL), technology mapping, and finally physical layout involving place-and-route. A mark applied at one level transfers down to lower levels. However, because a mark is nonfunctional, it may be removed by reverse engineering a design to a higher level in the design flow than that where the mark was applied. However, FPGA vendors generally believe that it is difficult to reverse engineer their devices, and they promise their customers that they will keep the bitstream specification confidential in order to raise the bar for reverse engineering [23]. Ken Hodor, product-marketing manager at Actel, claims that "antifuse-based FPGAs are by far the hardest device to reverse engineer" [8]. The SRAM-based 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 irregular row and column pattern due to the hierarchical interconnect network increases the complexity.

2.2 Vulnerability to Statistical Analysis

Although the new IPP approaches protect against mark removal through repetition-based statistical analysis, other statistic investigations could be launched. A naive approach to encoding a watermark would involve making direct use of symbols from a known alphabet or strings from a known language (e.g. ASCII encoding of words from a romance language). This approach would result in a frequency distribution of symbols that is likely to be quite different from that typically found in LUTs that are used to implement digital logic. An engineer could detect a watermark through statistical analysis given a large enough sample of typical digital designs. The degree of risk is directly proportional to the size of the watermark (i.e. the sample size for comparison). While this approach could be used to identify the likely existence of a watermark, it cannot be used directly to identify its location.

This problem can be attacked through three methods. First, the number of symbols selected from the alphabet can be reduced – the number of watermark bits can be made shorter. This technique is in direct opposition to the goal of achieving high confidence of verification by watermarking many bits, and thus we rejected it. Second, a mapping function can be produced that will translate the symbols in the watermark alphabet into appropriate symbols from the typical design distribution – thus giving the watermark a statistical signature closer to a typical design. We do not currently have a large enough set of complete designs to be able to characterize the typical distribution, and thus producing such a mapping function is problematic. We have chosen to implement the third approach, which whitens the spectrum of the watermark making it not look like any particular spectrum. Thus, while it will still be possible to find a mark by asking "does this look like a typical design?" (given enough information regarding typical design characteristics) it is not possible to find the mark by asking "is there a mark in English?" As mentioned earlier this spectral whitening primarily is achieved through the application of secure hash functions.

2.3 Verifying Altered Designs

As discussed in Section 1.4, watermarking places ownership marks in specific LUTs based on the design seed and desired marks. Therefore, knowing the design seed and embedded marks, the location of each mark is known to the owner. Verification then only requires comparisons to be made between the presumed mark and the LUT bits. However, if the design has been altered (e.g. the bitstream reverse engineered to the physical design layout level and rearranged using a design or floorplan editor), the mark may have been moved to a different LUT location. If such a case arises, mark extraction for ownership or source verification involves blindly searching LUTs for the multiple ownership watermarks, much in the

same manner as the ownership portion of fingerprints is extracted (see Section 1.4). This is unfortunately more complex and less convincing than extracting the marks from known locations.

3 Related Work

Many current IP protection techniques are based on encrypted source files. For example, encrypted HDL modules disguise the form and structure from IP users. This allows the IP users the ability to incorporate soft modules and high performance simulation models into their design using a CAD tool provided with the decryption key, without exposing the IP to theft. However, this approach has been routinely and successfully attacked, often by directly attacking the CAD tool. Therefore, there is no foreseeable IP protection technique based on encrypted source files, despite stronger forms of encryption and more thorough systems engineering.

Signature hiding techniques for image, video, and audio signals have recently received a great deal of attention. Digital image steganography has been especially well explored [4,21,26]. Although many mark security and verification issues have been raised [5], several image watermarking techniques do exist that have been shown to be robust against all known attacks [22]. Digital audio protection has proven to be even more difficult, but many different techniques have nevertheless been proposed [1,2,4]. Video stream protection techniques have also been developed [9,20].

Techniques have arisen that provide general intellectual property protection through watermarking. Marks are embedded at the behavioral level down to the physical layout by imposing design constraints [3,10,12,13]. A different set of synthesis and optimization issues arises when applying marks at different design phases (physical synthesis of FPGA-based design vs. behavioral synthesis). Addressing the design at a lower level of abstraction provides the advantage of a larger design space and greater flexibility, making it possible to embed signatures that are significantly more difficult to detect and remove.

Cryptography is used for selecting a subset of FPGA physical design constraints for mark embedding, as it provides probabilistic randomization and therefore protection from added constraints. For this task, we use the standard cryptography tools from the PGP-cryptography suite, the secure hash function MD5, and the RSA/MIT stream cipher RC4 [19].

4 Approach

4.1 Global Flow

While the basic design flows for watermarking and fingerprinting are no different than described in [16] and [14] respectively, the techniques introduced here create new approaches for three sub-functions: mark preparation, mark embedding, and mark verification. The pseudo-code and explanations in Sections 4.1.1 and 4.1.2 represent the global flow of each technique.

4.1.1 Watermarking

```
1.  create initial non-watermarked design;
2.  extract timing and area information;
3.  prepare marks;
4.  establish mark locations;
5.  modify netlist and physical constraints for mark locations;
6.  execute vendor place-and-route tools on modified netlist;
7.  embed marks;
8.  incorporate unused logic blocks into design;
9.  if !(meet timing criteria) {
10.         retry with fewer marks, else terminate with success;
11. }
```

Steps 1 and 2 are a part of any digital design flow. The original netlist is mapped, processes by place-and-route tools, and subjected to timing and area analysis. This timing and area information is later used for calculating the overhead incurred due to watermarking. Ownership mark preparation is then performed in Step 3, and mark locations are defined by a design seed in Step 4. The physical constraints based on the established mark locations are input to the netlist and CAD tool constraints file in Step 5, allowing the modified design to be re-mapped and re-processed by the place-and-route tool in Step 6. Steps 7 and 8 embed the marks in the appropriate LUTs which are incorporated into the rest of the design by receiving dummy inputs and outputting to neighboring "don't care" inputs, further hiding the marks. Timing analysis is done in Step 9, establishing the timing overhead incurred due to watermarking. If the overhead is deemed unacceptable, the process is repeated with fewer marks, which also changes the mark locations determined in Step 4.

4.1.2 Fingerprinting

```
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 logic blocks into design;
11.                 store instance;
12. }       }       }       }
13. prepare ownership marks;
14. for (i=1;i<=# of recipients;i++) {
15.     prepare recipient marks(i);
16.     select tile instances from database;
17.     embed marks;
18. }
```

As in watermarking, Steps 1 and 2 create the physical layout for the non-fingerprinted design, establishing the basis for all area and timing overhead. Steps 3-12 perform the design partitioning (tiling), creating a database of tile instances. Step 13 prepares the ownership marks, and Steps 14-18 are executed for each distributed instance of the design. Fingerprinting Steps 10, 13, and 17 are analogous to watermarking Steps 8, 3, and 7. However, the locations of fingerprint marks are also a function of tile instances, not just the design seed.

4.2 Mark Preparation

The improved method of mark preparation is the first major diversion from the original watermarking and fingerprinting techniques. Mark preparation now has the specific focus of creating small and multiple marks. One such advancement in efficiency is that the small and multiple marks make error-correction coding (ECC) block interleaving unnecessary. The marks are small enough that interleaving blocks would not add much value. However by increasing the number of marks, verification (the original motivation for interleaving due to the singular mark) will not be affected.

The marks to be embedded originate as 7-bit ASCII strings that can be printed using traditional I/O mechanisms[2]. Their sizes are limited by the subsequent hash function specifications. The mark strings are given to the watermarking system for embedding in the circuit and are later produced by the

[2] A set of marks indicating design instance recipient information must be included for fingerprinting.

verification program. The marks are transformed via a hash function (i.e. MD5), creating marks each capable of fitting in a single LUT[3], while still incorporating the user-defined number of ECC bits as discussed below. This step is crucial to the enhancements enabled by small and multiple marks, as the original watermarking and fingerprinting techniques prepared one large signature. As a consequence of the hash function, the marks are whitened so as not to look like any particular statistical spectrum, as described in Section 2.2. This whitening of the signal does not mask its content but rather its existence. By making the mark have a flat distribution, the marks will be more difficult to detect.

Finally, mark preparation involves adding ECC, which helps combat attempts to modify or remove the marks by changing LUT bits. If the modification is small enough, ECC codes help increase the possibility of retrieving the original marks and, if successful, provide proof of design tampering. A tradeoff exists between the number of bits allocated to each mark and to ECC, as the sum must not exceed the size of the target architecture's LUTs while still providing enough tampering protection through ECC.

4.3 Mark Embedding

As mentioned in Section 4.1.2, the process of selecting the mark locations is different for watermarking and fingerprinting. Watermarking locations are determined by a secure function and a seed, which leaves a different design with the same set of marks still secure. This additional security is relevant to mark verification efficiency discussed in Sections 1.4 and 4.4.

Fingerprinting locations are independent of hash functions, as they are determined by tile instances. Each tile instance is generated by placing the unused LUTs 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 design instance are unique, which has a negative affect on mark verification efficiency. The one constant that exists is that the design recipient marks are always kept in the same location relative to the ownership marks (the same logic block if the target architecture's logic blocks contain multiple LUTs), thus simplifying the process of fingerprint retrieval.

After the mark locations are determined, the process of mark embedding begins. As in the previous watermarking and fingerprinting techniques, the LUTs to be implanted with the marks are given arbitrary inputs and outputs in order to further disguise the marks. The inputs are simply taps off of passing signals, and the outputs are routes to neighboring logic block "don't care" inputs. This incorporation helps to hide the marked LUTs without severely impacting design performance, as the dummy outputs are not a functional design component.

Finally, the marks themselves are embedded in the predefined LUTs by reprogramming the respective bits in the bitstream. Therefore, the final step of mark embedding is an entirely post-processing step.

4.4 Mark Verification

When the suspicion of theft or misappropriation arises, an unbiased verification team is presented the configuration in question. For watermarking, the IP vendor must produce the design seed that they claim was used to produce the block and upon which the mark locations are based. The verification team uses the seed and reverses the signature preparation and embedding process by first identifying the LUTs used for hiding the marks. Once the marks are extracted and, if necessary, the ECC is applied, the marks are decrypted using a known key and hash function. Finally, the original ASCII signature is revealed, and if the signature identifies the IP vendor, ownership has been established. As discussed in Section 1.4, ownership can also be publicly proved by revealing a subset of the multiple watermarks found in the design without creating the possibility that other design (the same design or other designs containing the same mark set) recipients remove the ownership information.

---

[3] If the target architecture possesses logic blocks with multiple LUTs, a mark can transcend a single LUT to fit within a logic block's available LUTs.

The process is slightly different for fingerprinting, as the mark locations are unknown. The IP vendor must produce a subset of the ownership marks that they claim to be embedded in the design, but the verification process is reversed. The signatures are encrypted using a known key and are applied to the secure hash function and ECC creating the bits used to code the LUTs. The verification team then searches the LUTs for the subset of bits. If all or a large percentage of the subset is found, ownership has been established. Once the locations of the ownership marks are known, the watermarking extraction process can be implemented on instance recipient mark retrieval, as recipient marks are always in a constant location relative to the ownership marks. The relevant LUT bits are decrypted using the known key and printed out. The result is then used to establish the source of theft or misappropriation.

5 Experimental Results

5.1 Objectives

Experiments have been used to evaluate the overhead (area and timing) of the proposed watermarking and fingerprinting approaches as well as the fingerprint security against recipient collusion.

When calculating area overhead for the proposed techniques, it must be noted that place-and-route tools rarely pack utilized logic blocks, and therefore LUTs, into a minimal area. Unused logic introduces flexibility into the place-and-route step that may be essential for completion or good performance. However, these unused LUTs can be used for embedding marks, but should not be considered area overhead. For example, an initial design may possess a region that contains 100 utilized logic blocks but also 15 unutilized logic blocks. Area overhead should not include those 15 blocks. Rather, area overhead must be calculated as the area used by the watermarked design minus the total area of the original design, including unused logic blocks and LUTs. In addition, the constraints imposed on placement for mark embedding may also contribute to timing overhead, as a marked LUT may require that a critical path be lengthened.

The fingerprinting approach incurs design effort overhead and additional area and timing overhead due to tiling. Mark security is also an issue in fingerprinting, as recipient collusion could remove parts of the instance recipient marks.

The smaller and multiple marks technique does not have an impact on the area, timing, or design effort overhead required for the existing FPGA IPP techniques. A given number of LUT bits and unused LUT locations are available, and all of the bits may be used to encode several smaller marks where one large watermark had been. For example, one 320-bit mark distributed over twenty LUTs could be transformed to twenty 16-bit marks without affecting the design.

5.2 Designs

To evaluate the area and timing overhead of the watermarking approach, we conducted an experiment on three large real-world designs: a MIPS R2000 processor core designed for FPGAs [11], a reconfigurable Automatic Target Recognition (ATR) system [25], and a digital encryption standard (DES) design [17]. The MIPS core and the DES design were both implemented on the Xilinx XC4028EX-3-PG299, and the ATR system was implemented on the XC4062XL-3-PG475. For each design, the smallest possible device was used. For fingerprinting overhead evaluation, experiments were performed on nine MCNC designs, each of which was implemented on the XC4028EX-3-PG299. In Step 2 of both the watermarking and fingerprinting pseudo-code in Sections 4.1.1 and 4.1.2, the number of unused LUTs was calculated and the circuit timing was noted. The original area and timing statistics for both sets of designs are displayed in Table 1.

design	# used LBs	# spare LBs	min period (ns)
MIPS R2000	756	268	185.0
ATR	1876	214	424.5
DES	875	149	166.3
9sym	46	49	71.6
c499	94	96	104.9
c880	110	115	110.8
duke2	93	100	87.9
rd84	27	28	50.2
planet1	95	100	145.0
s9234	195	206	135.0
sand	82	90	97.6
styr	78	81	150.6

Table 1. Original physical layout statistics

5.3 Watermarking Results

Experimental results reveal that the new watermarking approach does not require more area or timing overhead that the existing FPGA watermarking technique. Due to the place-and-route tool not packing the logic to maximum density, there is essentially no area overhead required by the new approach. As expected, the unused logic was used to embed the marks and therefore increased the density of utilized logic blocks and LUTs. If place-and-route tools packed logic to a higher density, a certain degree of area overhead may become apparent.

Following the pseudo-code approach detailed in Section 4.1.1, location constraints were placed on each design before place-and-route. An iteratively larger number of marks (until all unused LUTs were filled) were embedded, and the circuit timing was noted and compared to the original design. The results are shown in Tables 2-4.

# 16-bit marks	50	98	162	200	242	288	338	392	450	512
% resources	3.31	6.48	10.71	13.23	16.01	19.05	22.35	25.93	29.76	33.86
% timing	-1.04	-0.47	3.17	-7.15	-4.69	1.65	-11.53	2.47	11.95	-5.23

Table 2. MIPS R2000 – Impact of number of 16-bit marks on resources and speed

# 16-bit marks	2	50	98	184	288	374	428
% resources	0.05	1.33	2.61	4.90	7.68	9.97	11.41
% timing	-10.74	3.46	-25.93	-7.99	-13.50	10.25	-1.57

Table 3. ATR - Impact of number of 16-bit marks on resources and speed

# 16-bit marks	2	50	98	158	200	242	298
% resources	0.11	2.86	5.60	9.03	11.43	13.83	17.03
% timing	-22.98	-14.83	-5.07	-1.90	11.05	-11.93	-3.28

Table 4. DES - Impact of number of 16-bit marks on resources and speed

For each table, the top two rows show the number of the 16-bit (Xilinx 4000 LUT size) marks. The next two rows show the area increase and timing degradation. As mentioned above, the area overhead is nearly 0%, but the additional percentages of utilized logic blocks are noted in the tables. For timing degradation, positive percentages indicate a decrease in performance. Therefore, some marked designs recorded a timing improvement. This can be explained by the dramatically different placement and corresponding timing that often results from relatively small design changes. The timing impact of watermarking is below the characteristic variance associated with such small changes. Therefore, timing degradation is non-monotonic with the number of marks.

Figures 1 and 2 represent two iterations of the watermark experiment process. Figure 1 is the original layout of the DES design with no watermarking constraints. The area and timing statistics for the design are noted in Table 1. Note that optimal logic density for the original placement does not exist, as many unused logic blocks are dispersed throughout the design. The experimental process continued until all unused LUTs contained a mark. Figure 2 shows the final layout with 298 16-bit marks and, therefore, the maximum amount of location constraints. The location of each LUT containing a mark is hidden by the incorporation of each LUT into the design. Inputs are taken from passing signals, and the outputs are routed to neighboring logic block "don't care" inputs. Comparisons between Figures 1 and 2 reveal that area overhead is negligible (only logic density is increased), and Table 4 indicates that the timing overhead is actually negative for this iteration of mark embedding.

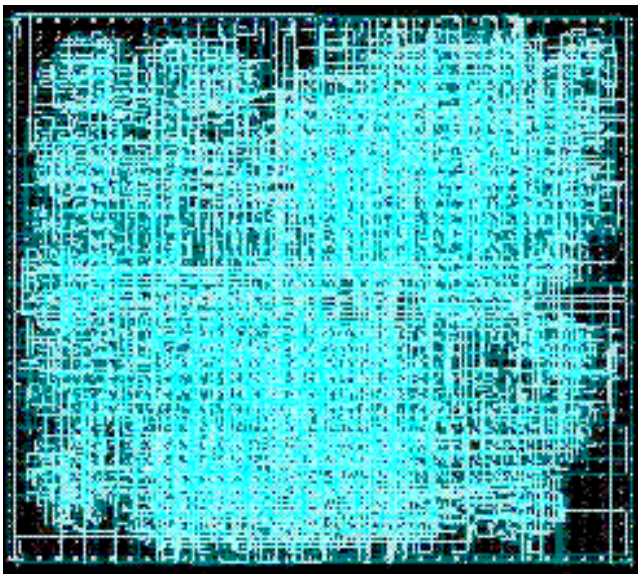

Figure 1. DES original layout

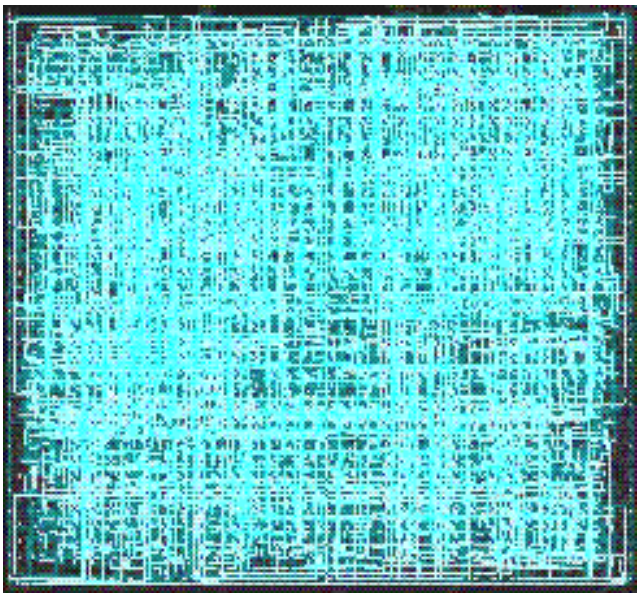

Figure 2. DES with 298 16-bit marks

5.4 Fingerprinting Results

Fingerprinting area overhead due to tiling is shown in Table 5. If more marks are needed, additional resources could be added to the design.

Design	# 16-bit marks	% resources
9sym	16	.065
c499	26	.021
c880	36	.045
duke2	26	.075
rd84	8	.037
planet1	28	.053
s9234	46	.056
Sand	26	.098
Styr	20	.038

Table 5. Area overhead for number of 16-bit marks

Fingerprinting timing overhead, which is variable and user specified, is show in Figure 3. Variability is achieved by simply discarding tile instances that do not meet defined timing specifications. Therefore, Figure 3 displays timing overhead in terms of instance yield (i.e. number of tile instances that meet the timing specifications / total number of tile instances) for certain timing specifications (measured as percent increase over the original, non-fingerprinted design timing). Results indicate that if specifications consider a 20% timing degradation acceptable, then approximately 90% of total tile instances will be acceptable. As with watermarking, timing degradation due to fingerprinting appears to be below the characteristic variance associated with small design changes.

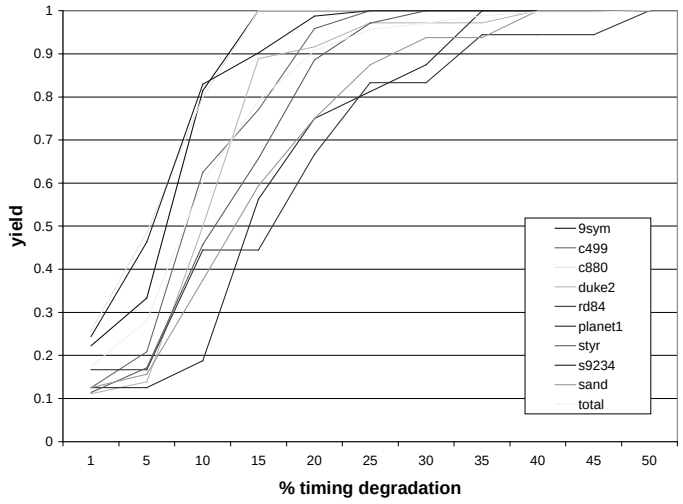

Figure 3. Instance yield vs. timing specifications

Fingerprint mark security against comparison collusion is detailed in [14]. Security increases with the size of the tile, but back-end CAD tool effort required for tile instance generation also increases with the size of the tile. Figure 4 displays the chance that a certain percentage (graphs in 10% increments) of design instance recipient information could be removed by a growing number of colluders (tile size of 40 logic blocks). The results show that even for small tile sizes, the likelihood that a colluding group could remove even a small number of marks is quite remote. For example, 15 colluders would have one chance

in approximately 4 million to remove 30% of the marks by comparison collusion. The chance would be even more remote if larger tile sizes were used, but design effort would increase[4].

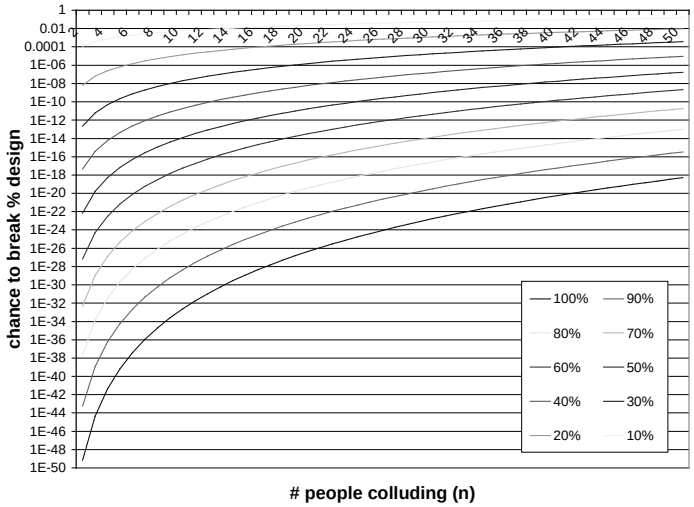

Figure 4. Chance to remove % of recipient marks (tile size = 40)

The tradeoff between design effort and fingerprint security is important to consider, as design effort may often be the limiting factor for tile size selection. Figure 5 is a direct multiplication comparison between fingerprint security and design effort.

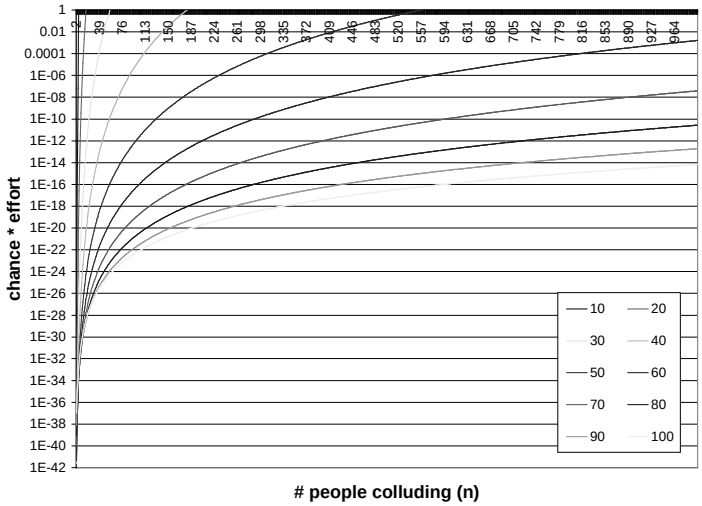

Figure 5. Chance to remove all recipient marks * design effort

Alternate comparisons between security and effort (e.g. security * effort^2) may be more accurate for certain applications. High security against large colluding groups is not necessarily required for many of today's applications, as a large number of design instances are not normally easily available. Also, modern FPGA mapping technology is time consuming, so current design settings place a strong emphasis on design effort.

[4] For fingerprint design effort and security, the following constants were assumed: device size = 400 logic blocks, number of unused logic blocks/number of total logic blocks = 0.1, and instance yield (due to timing specifications) = 0.9.

6 Conclusion

We have introduced new FPGA design watermarking and fingerprinting techniques for IPP that are more efficient for copy detection, more convincing for ownership and recipient verification, and more secure and robust against mark removal than existing techniques. These improvements are achieved without increasing user design effort, CAD tool effort, or area and timing overhead.

Acknowledgements

This work was supported by the Defense Advanced Research Projects Agency of the United States of America, under contract F30602-96-C-0350 and subcontract QS5200 from Sanders, a Lockheed Martin company.

References

[1] W. Bender et al., "Techniques for Data Hiding," *IBM Systems Journal*, vol. 35, no 3-4, 1996, 313-336.

[2] L. Boney et al., "Digital Watermarks for Audio Signals," *International Conference on Multimedia Computing and Systems*, 1996.

[3] E. Charbon, "Hierarchical Watermarking in IC Design," *Custom Integrated Circuits Conference*, 1998.

[4] I.J. Cox et al., "Secure Spread Spectrum Watermarking for Images, Audio, and Video," *International Conference on Image Processing*, 1996.

[5] S. Craver et al., "Can Invisible Watermarks Resolve Rightful Ownership?" *Storage and Retrieval for Image and Video Databases, Proceedings of the SPIE*, vol. 3022, 1997, 310-321.

[6] W. Diffie and M. Hellman, "New Directions on Cryptography," *IEEE Transactions on Information Theory*, vol. IT-22, no. 6, Nov. 1976, 644-654.

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

[8] R. Goering, "IP98 Forum Exposes Struggling Industry – Undefined Business Models, Unstable Core Prices Cited," *EE Times*, Issue 1000, March 30, 1998.

[9] F. Hartung and B. Girod, "Copyright Protection in Video Delivery Networks by Watermarking of Pre-Compressed Video," *ECMAST'97, Springer Lecture Notes in Computer Science*, vol. 1242, 1997, 423-436.

[10] I. Hong and M. Potkonjak, "Behavioral Synthesis Techniques for Intellectual Property Protection," unpublished manuscript, 1997.

[11] B. Hutchings et al., *BYUcore: A MIPS R2000 Processor for FPGAs*, 1997.

[12] A.B. Kahng et al., "Robust IP Watermarking Methodologies for Physical Design," *Design Automation Conference*, 1998, 782-787.

[13] A.B. Kahng et al., "Watermarking Techniques for Intellectual Property Protection," *Design Automation Conference*, 1998, 776-781.

[14] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "Fingerprinting Digital Circuits on Programmable Hardware," *International Workshop on Information Hiding*, 1998, 16-31.

[15] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "Low Overhead Fault-Tolerant FPGA Systems," *IEEE Transactions on VLSI*, vol. 6, no. 2, June 1998, 212-221.

[16] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "Signature Hiding Techniques for FPGA Intellectual Property Protection," *International Conference on Computer-Aided Design*, 1998.

[17] J. Leonard and W. H. Mangione-Smith, "A Case Study of Partially Evaluated Hardware Circuits:

Key-Specific DES," *Field Programmable Logic*, 1997, 151-160.

[18] J. Montanaro et al., “A 160MHz 32b 0.5W CMOS RISC Microprocessor,” *IEEE Journal of Solid-State Circuits*, vol. 31, no. 11, Nov. 1996, 1703-1714.

[19] B. Schneier, *1963- Applied Cryptography: Protocols, Algorithms, and Source Code in C*, New York: John Wiley & Sons, 1996.

[20] 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.

[21] M.D. Swanson et al., "Transparent Robust Image Watermarking," *International Conference on Image Processing*, 1996.

[22] A.H. Tewfik and M. Swanson, "Data Hiding for Multimedia Personalization, Interaction, and Protection," *IEEE Signal Processing Magazine*, 1997, p. 41-44.

[23] S. Trimberger, Personal Communication, Xilinx Corporation, 1997.

[24] J. Turley, “ARM Grabs Embedded Speed Lead,” *Microprocessor Report*, vol. 10, 1996.

[25] J. Villasenor et al., "Configurable Computing Solutions for Automatic Target Recognition," *Proceedings of IEEE Workshop on FPGAs for Custom Computing Machines*, 1996, 70-79.

[26] R.B. Wolfgang and E.J. Delp, "A Watermark for Digital Images," *Applications of Toral Automorphisms*, vol. 3, 1996, 219-222.

[27] Xilinx, The Programmable Logic Data Book, San Jose, CA, 1996.