Open access • Proceedings Article • DOI:10.1145/309847.310080

Robust FPGA intellectual property protection through multiple small watermarks

John Lach, William H. Mangione-Smith, Miodrag Potkonjak

Institutions: University of California, Los Angeles

Published on: 01 Jun 1999 - Design Automation Conference

Topics: Digital watermarking, Copy protection and Hash function

Robust FPGA Intellectual Property Protection Through Multiple Small Watermarks

John Lach
UCLA EE Department
56-125B Engineering IV
Los Angeles, CA 90095
310-794-1630
jlach@icsl.ucla.edu

William H. Mangione-Smith
UCLA EE Department
56-125B Engineering IV
Los Angeles, CA 90095
310-206-4195
billms@ee.ucla.edu

Miodrag Potkonjak
UCLA CS Department
3532G Boelter Hall
Los Angeles, CA 90095
310-825-0790
miodrag@cs.ucla.edu

ABSTRACT

A number of researchers have proposed using digital marks to provide ownership 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 field programmable gate array (FPGA) intellectual property (IP) that achieves robustness by responding to these three weaknesses. The key technique involves using secure hash functions to generate and embed multiple small marks that are more detectable, verifiable, and secure than existing IP protection techniques.

Keywords

Field programmable gate array (FPGA), intellectual property protection, watermarking

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 [7,17,22]. 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 (e.g. Lexra, Altera, Xilinx, VA Research) 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.

An existing FPGA design watermarking [15] technique attempts 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 [14]) and specific design instance recipient (fingerprinting only). This existing FPGA IP protection technique 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 technique enables efficient copy detection for identifying stolen or misappropriated IP, even if it is embedded in a complex system. The technique also provides simple mark extraction and convincing design source 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 technique provides 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 technique.

1.2 Existing FPGA Watermarking Technique

FPGA design watermarking embeds a digital mark in unused lookup tables (LUTs) throughout the design [15]. 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.

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, the existing FPGA watermarking technique restricts mark placement to logic block LUTs, and it can be quickly established where LUTs are located in an FPGA bitstream. (Bitstreams are attained by monitoring the serial line between the FPGA and the bitstream storing EPROM during configuration.) Therefore, LUTs can be scanned for ownership marks, making copy detection reasonably efficient.

The new FPGA watermarking technique allows for even more efficient copy detection than the existing technique provides 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 (the existing FPGA watermarking technique uses 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 multiple 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 technique eliminates 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 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. Even without mark location information (e.g. when the IP is merged into a larger design) it is possible to detect and verify the mark due to the searching efficiency the small and multiple mark approach provides.

1.5 Mark Security and Robustness

Our new technique embeds multiple small marks, as opposed to a single large mark, creating greater mark security and robustness than the existing technique provides.

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.

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 a new FPGA design watermarking technique for IPP that provides 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 this new IPP technique. Section 4 details the new approach to FPGA IPP, and Section 5 evaluates the new approach. 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 technique. 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, but 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 [S. Trimberger, Xilinx Corporation, personal communication]. 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 approach protects 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 a 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) or merged with other IP, 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. This is unfortunately more complex and less convincing than extracting the marks from known locations. However, it is important to note the position independent nature of the watermark, as the complexity of blindly searching the LUTs is greatly reduced by the use of small multiple marks.

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 [4,20,24]. 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 [21]. 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 [9,19].

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 [18].

4. APPROACH

4.1 Global Flow

While the basic design flow for watermarking is no different than described in [15], the technique introduced here creates new approaches for three sub-functions: mark preparation, mark embedding, and mark verification. The pseudo-code and explanations below represent the global flow of the technique.

```
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, processed by the 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.

4.2 Mark Preparation

The improved method of mark preparation is the first major diversion from the original watermarking technique. 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. 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 verification program. The marks are transformed via a hash function (i.e. MD5), creating marks each capable of fitting in a single LUT[1], 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 technique 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

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.

After the mark locations are determined, the process of mark embedding begins. As in the previous watermarking technique, 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. 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 proven 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.

5. EXPERIMENTAL RESULTS

5.1 Objectives

Experiments have been used to evaluate the overhead (area and timing) of the proposed watermarking approach. When calculating area overhead, 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 smaller and multiple marks technique does not have an impact on the area, timing, or design effort overhead required for the existing FPGA watermarking technique. 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

[1] 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.

[23], and a digital encryption standard (DES) design [16]. 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. In Step 2 of the pseudo-code in Section 4.1, the number of unused LUTs was calculated and the circuit timing was noted. The original area and timing statistics 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

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 than 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, 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.

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.

# 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

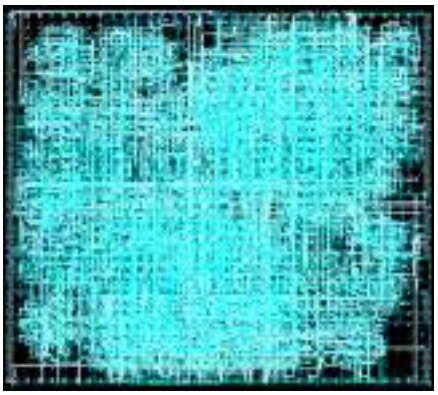

Figure 1. DES original layout

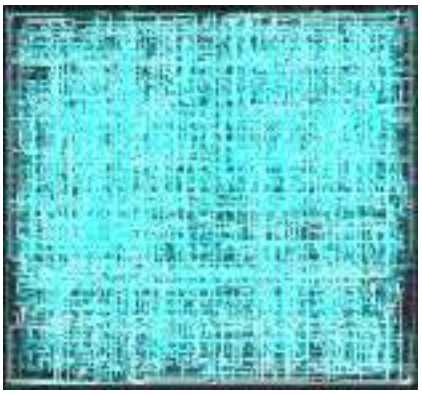

Figure 2. DES with 298 16-bit marks

6. CONCLUSION

We have introduced new FPGA design watermarking technique for IPP that is 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 Air Force Research Laboratory 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, 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," *Design Automation Conference*, 1999.

[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, "Signature Hiding Techniques for FPGA Intellectual Property Protection," *International Conference on Computer-Aided Design*, 1998.

[16] J. Leonard and W. H. Mangione-Smith, "A Case Study of Partially Evaluated Hardware Circuits: Key-Specific DES," *Field Programmable Logic*, 1997, 151-160.

[17] 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.

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

[19] 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.

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

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

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

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

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

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