INTELLECTUAL PROPERTY PROTECTION BY WATERMARKING COMBINATIONAL LOGIC SYNTHESIS SOLUTIONS

Darko Kirovski, Yean-Yow Hwang, Miodrag Potkonjak, and Jason Cong
Computer Science Department, University of California, Los Angeles

ABSTRACT

The intellectual property (IP) business model is vulnerable to a number of potentially devastating obstructions, such as misappropriation and intellectual property fraud. We propose a new method for IP protection (IPP) which facilitates design watermarking at the combinational logic synthesis level. We developed protocols for embedding designer- and/or tool-specific information into a logic network while performing multi-level logic minimization and technology mapping. We demonstrate that the difficulty of erasing author's signature or finding another signature in the synthesized design can be made arbitrarily computationally difficult. We also developed a statistical method which enables us to establish the strength of the proof of authorship. The watermarking method has been tested on a standard set of real-life benchmarks where exceptionally high probability of authorship has been achieved with negligible overhead in solution quality.

1. INTRODUCTION

The complexity of modern system synthesis as well as shortened time-to-market requirement has resulted in design reuse as a predominant system development paradigm. The new core development strategies have affected the business model of virtually all CAD and semiconductor companies. To overcome the difficulties in core-based system design, the VSI Alliance has identified six technologies crucial for enabling effective design reuse: system verification, mixed signal design integration, standardized on-chip bus, manufacturing related test, system-level design, and intellectual property protection (IPP) [VSI97].

We have developed the first approach for IPP which facilitates design watermarking at the combinational logic synthesis level. The watermark, a designer- and/or tool-specific information, is embedded into the logic network of a design at a preprocessing step. The watermark is encoded as a set of design constraints which do not exist in the original specification. The constraints are uniquely dependent upon author's signature. Upon imposing these constraints to the original logic network, a new input is generated which has the same functionality and contains user-specific information. The added constraints result in a trade-off. The more additional constraints, the stronger the proof of authorship, but the higher overhead in terms of quality of the synthesis solution. However, the application of the synthesis algorithm results in a solution which satisfies both the original and constrained input. Proof of authorship is based upon the fact that the likelihood that another application returns a solution to both the original and constrained input is exceptionally small. The developed watermarking technique is transparent to the synthesis step and can be used with any logic synthesis tool. We demonstrate that the developed IPP approach can be used to:

- *Prove authorship of the design at levels of abstraction equal or lower than logic synthesis.* Existence of a user-specific signature in the solution of a multi-level optimization or technology mapping problem clearly identifies the author of the input design specification (initial input logic network).
- *Protect the synthesis tool.* The signature of the tool developer, embedded in logic synthesis solutions, clearly indicates the origin of the synthesis tool.

2. RELATED WORK

Watermarking of artifacts has received great attention in the research community. A variety of techniques have been proposed for hiding data in images [Ber96, Cox96], audio [Ben96, Cox96], text [Ber96] and video [Har97]. Protocols for watermarking active IP have been developed at the physical layout [Kah98] and behavioral specification [Hon98] level. A routing-level approach for fingerprinting FPGA designs has been introduced [Lac98]. Embedding signatures into a design at the logic synthesis level has advantages over corresponding efforts at the higher and lower levels of the design process. Firstly, watermarking a behavioral specification often does not exhibit sufficient potential for embedding large signatures which are crucial for high authorship credibility. Physical layout should not be an exclusive domain for IPP because in that case only the solution to the physical design is protected.

Combinational logic synthesis has been extensively studied. A good survey of minimization techniques and existing synthesis frameworks is presented in [DeM94, Hac96]. Recent improvements in combinational logic synthesis include refined covering algorithms [Lia97], optimizations on networks described using black boxes [Liu97], power optimization [Tiw96], etc. In addition, logic synthesis for FPGAs has been a very active research area (see survey in [Con96a]).

3. WATERMARKING DESIDERATA

The recently proposed Strawman initiative [VSI97] of the Development Working group on IPP calls for the following desiderata for techniques which act as deterrents in order to properly ensure the rights of the original designers.

- **Functionality Preservation.** Design specific functional and timing requirements should not be altered by the application of IPP tools.
- **Minimal Hassle.** The technique should be fully transparent to already complex design and verification process.
- **Minimal Cost.** Both the cost of applying the protection technique and its hardware overhead should be as low as possible.

- **Enforceability.** The technique should provide strong and undeniable proof of authorship.
- **Flexibility.** The technique should enable a spectrum of protection levels which correspond to variable cost overheads.
- **Persistence.** The removal of the watermark should result in a task of the difficulty equal to the complete redesign of the specified functionality.

In addition to the stated VSI intellectual protection requirements, our approach also provides proportional protection of all parts of the design.

4. WATERMARKING LOGIC SYNTHESIS

The synthesis flow which employs watermarking of combinational logic synthesis solutions encompasses several phases illustrated in Figure 1. The first three phases in the watermarking approach are the same for both multi-level logic minimization and technology mapping.

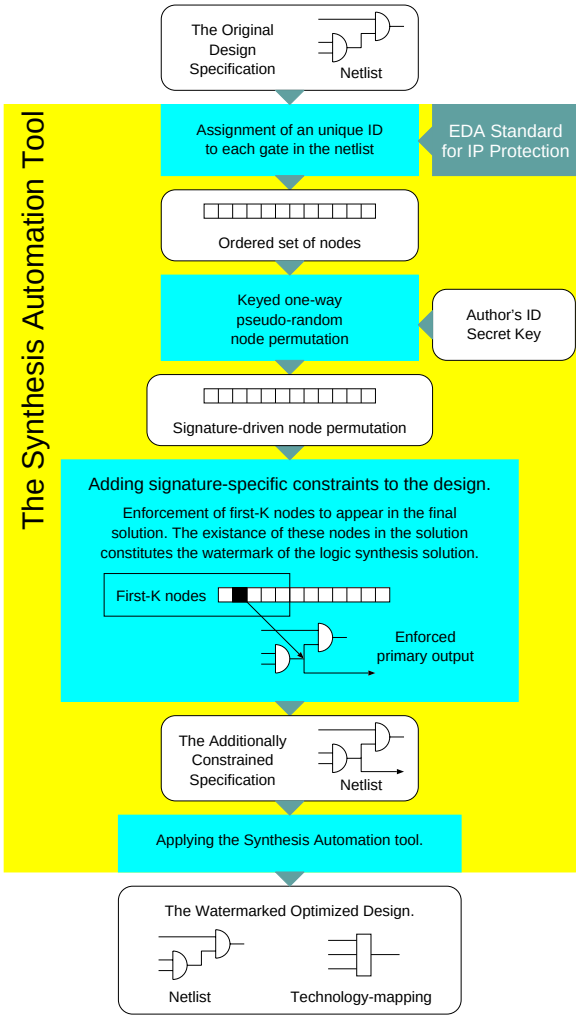

Figure 1: The protocol for hiding information in solutions for multi-level logic optimization and technology mapping.

In the first step, to ensure that the watermark cannot be misinterpreted, the gates in the initial logic network specification are sorted using an industry standard. As a result of this procedure, each gate of a given logic network can be assigned a single identifier which is unique with respect to the identifiers assigned to gates in the remainder of the network. Next, K gates are selected in a way specific to the designer's or tool developer's signature. We use a keyed RSA one-way function to generate pseudo-random bits [Men97] which guide the process of iterative gate selection. The outputs of the selected gates are explicitly assigned to become primary outputs. We have applied this protocol to the technology mapping synthesis step. Although the same protocol can be applied to watermark multi-level logic minimization solutions, for this task we provide an alternative protocol. Initially, it also generates pseudo-primary outputs according to the user's signature, and, in addition, uses them as inputs into an additional logic network which is embedded into the initial design specification. The protocol builds the embedded network according to the designer's or tool developer's signature.

After additionally constraining the initial design specification, the optimization algorithms are applied to the constrained logic network. The result retrieved by the synthesis algorithm satisfies both the initial and constrained design specification. The proof of authorship is dependent upon the likelihood that some other algorithm, when applied to the initial input, retrieves solution which also satisfies the constrained input.

4.1. Gate Ordering

The watermarking process starts by assigning a unique identification number ID_i to each gate G_i from the set G of gates which are not used as primary outputs. The unique identification number ID_i is selected from the set $ID_i \in ID = \{1...N\}$ of N successive numbers, where N is the cardinality of the set G. We have two main goals in this step: to map the network into a linear array so that cryptographical tools can be directly applied and to develop a uniquely defined IPP procedure in such a way that the degrees of freedom for potential attackers are maximally reduced.

To avoid misinterpretation of this ordering, we propose that an industry standard has to be established. The network has to be numbered in such a way that any two nodes that have different functionality and different transitive fan-in and fan-out networks are assigned different IDs. However, finding whether two nodes are functionally and topologically identical is a hard problem. The special case of the problem of finding whether two networks are identical, when all gates perform equivalent functions, is equivalent to the graph isomorphism problem. This problem has been listed as open in terms of its complexity [Gar79]. Therefore, we propose a heuristic function that exploits the functional and timing properties of a node, to sort the nodes in a logic network. This function is explained using the pseudo-code in Figure 2. It performs iterative sorting of nodes, not used as primary outputs, using a list of criteria with distinct priorities. The objective of the ordering function is to partition a logic network $LN(G, C)$, where G is a set of nodes and C is a set of connections between nodes, into an ordered set M of node subsets $M_i \in G$ such that each subset contains exactly one node. We propose the following list of eight criteria for node identification:

C[1] The level LIN_i of node G_i with respect to the input. A node G_i has a level K if the longest path in the logic network from any input to G_i is of cardinality K.

C[2] The level $LOUT_i$ of node G_i with respect to the output. A node G_i has a level K if the longest path in the logic network from any output to G_i is of cardinality K.

C[3] Number of nodes in the transitive fan-in of G_i at level $K < LIN_i$.

C[4] Number of nodes in the transitive fan-out of G_i at level $K < LOUT_i$.

C[5] Functionality, fan-in, and fan-out of nodes in the transitive fan-in of G_i at level $K < LIN_i$.

C[6] Functionality, fan-in, and fan-out of nodes in the transitive fan-out of G_i at level $K < LOUT_i$.

C[7] Functionality, fan-in, and fan-out of the fan-in and fan-out of nodes in the transitive fan-in of G_i at level $K < LIN_i$.

C[8] Functionality, fan-in, and fan-out of the fan-in and fan-out of nodes in the transitive fan-in of G_i at level $K < LOUT_i$.

Given a logic network $LN = \{G_1, ..., G_N, C\}$ with a set $I = \{I_1, ..., I_K\}$ of inputs and set $O = \{O_1, ..., O_L\} \in G$ of output nodes.

Ordered set M of sets of nodes $M = \{M_0 = G - O\}$
For each Criteria Function $C[i], i = 1..8$
  For each set of nodes $M_i \in M$ with $|M_i| > 1$
    For each node $G_j \in M_i$ compute $G_j.objective = C[i](G_j)$
    Partition M_i into an ordered set of unordered sets $M_{i,P_1}, ..., M_{i,P_K}$ such that all $G_j \in M_{i,P_k}$ have the same $P_k = G_j.objective$ and $P_k > P_{k+1}$.
    Augment the new set of partitions into the initial set M in the following order ..., $M_{i-1}, M_{i,1}, ..., M_{i,K}, M_{i+1}, ...$
For each set of nodes $M_i \in M$ with $|M_i| > 1$
  Randomly partition M_i into an ordered set of unordered sets $M_{i,P_1}, ..., M_{i,P_K}$ each of cardinality equal to 1.
  Augment the new set of partitions into the initial set M in the following order ..., $M_{i-1}, M_{i,1}, ..., M_{i,K}, M_{i+1}, ...$

Figure 2: Proposed function for completely defined node ordering.

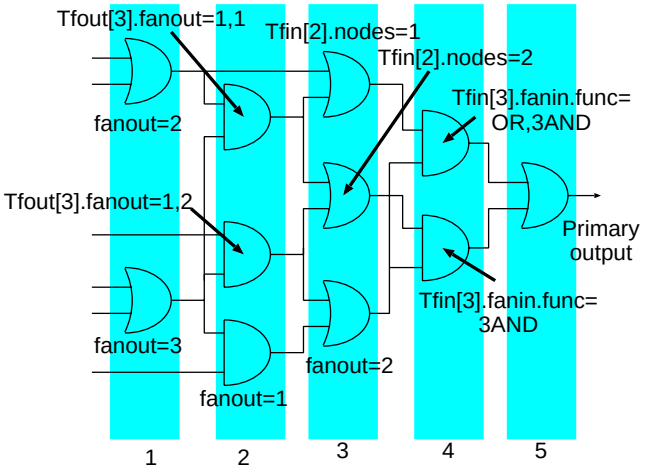

Figure 3: An example of ordering nodes according to the proposed set of sorting criteria.

An example how nodes are identified using the proposed set of sorting rules is given in Figure 3. Note that it is unlikely that two nodes have all parameters identical. This is due to the dependencies and non-symmetry between nodes in logic networks. If two nodes cannot be distinguished using the proposed set of rules, we assign random unique identifiers to these nodes and memorize the assignment for future proof of authorship.

4.2. Watermark Encoding and Embedding

In the next phase of watermarking, from the sorted set M of non-primary nodes, a subset $S \in M$ of cardinality $|S| = K$ is selected. The selection is pseudo-random and corresponds uniquely to the designer's or tool developer's signature. Next, each node in the selected subset S is explicitly added to the list of pseudo-primary outputs. By performing this step, the watermarking routine enforces nodes from the set S to be:

- visible in the final technology mapping solution.
- computed during the multi-level logic minimization of the logic network. Note that many subfunctions that exist in the input logic network do not exist in the optimized output logic network.

The node selection is performed in the following way. Since the node selection step of watermarking is not assumed to be the computation bottleneck, we use the RSA cryptographically secure pseudo-random bit-generator [Men97] to generate a sequence of bits which decides upon node selection. The keys used to drive the randomization process represent the user signature. The result of this phase is a pseudo-random signature-specific selection of a combination of K network nodes.

In the case of technology mapping of LUT-based FPGAs, the described node selection phase is the last phase in the protocol. However, it is important to stress the implications of a specific phenomenon in this problem. Cong and Ding [Con96a] have identified a class of MFFC nodes which are more likely to appear in the final solution than the remaining nodes. We have statistically evaluated the impact of this phenomenon on the strength of the proof of authorship enabled by our approach. For each instance of the problem, we have explicitly enumerated the ratio of MFFC nodes in the initial input specification (rin) and in the final solution ($rout$). We compute the likelihood of solution coincidence using the following formula: $p = (\frac{rout \cdot F}{rin \cdot T})^{rout \cdot W} \cdot (\frac{(1-rout) \cdot F}{(1-rin) \cdot T})^{(1-rout) \cdot W}$, where F is the number of non-primary gates in the final solution, T is the total number of non-primary gates in the initial logic network, and W is the number of gates pseudo-randomly selected to become pseudo-primary outputs during the watermarking phase.

The protocol described for technology mapping can be applied to watermark solutions to the multi-level logic minimization problem. However, we propose an alternative protocol which provides stronger proof of authorship due to embedded additional constraints. This protocol augments signature-specific constraints into the input logic network in two phases. In the first phase which is equivalent to the already described protocol for watermarking technology mapping solutions, the protocol marks the outputs of selected gates as visible by explicitly denoting them as pseudo-primary outputs. In the second phase, an additional network is augmented into the input. The additional network has as input variables the pseudo-primary output variables generated in the previous phase. The network is built according to the user's signature. The pseudo-code for building the additional signature is presented in Figure 4. The sequence of pseudo-random bits from the previous phase is used to provide a source of undeniable determination.

ILN is an input logic network.
PPN is an ordered set of pseudo primary nodes.
PRS = RSAbitGenerator(key1, key2), where $key1, key2$ are the designer and/or tool developer signature.
$ListAddedGates = null$
Repeat (Standard.cardinality_ of_ added_ gates)
  Gate G = Select(Library, PRS, Pointer)
  $AddedGates.Add(G)$
  $ILN.Add(G, G.fanin[Select(PPN, PRS, Pointer)$
  $PPN.Add(G.fanout)$
End Repeat
For each gate $G \in AddedGates$
  If G.fanout = null $ILN.fanout.Add(G)$

Figure 4: Proposed function for watermarking multi-level logic minimization solutions using network augmentation.

Using this sequence, firstly, a gate G from the available library of gates is selected. Then according to the pseudo-random sequence of bits, $G.fanin$ pseudo primary outputs are selected and used as inputs to the selected gate G. The output $G.fanout$ is added to the list of pseudo-primary outputs. This output is subject to selection in the future iterations of this procedure. This procedure can be infinitely repeated. A possible termination policy may be established using industry adopted standards.

The additionally constrained original input netlist is fetched to the optimization algorithm (multi-level logic minimization or technology mapping). The final solution is a network of cells (or subfunctions) which contains solution to the original problem and to the user-specific augmentation of the original problem. The proof of authorship relies on the difficulty to: modify the input in such a way that the pseudo primary outputs that correspond to the attacker's signature and the modified network that corresponds to the the attacker's key have a subsolution that is a subsolution to the initial problem watermarked with the designer's watermark.

4.3. Persistence to Possible Attacks

The attacker may try to modify the output locally in such a way that the watermark disappears or the proof of authorship is lowered bellow a predetermined standard. Therefore, the watermarking scheme has to be such that, to delete the watermark and still preserve solution quality, the attacker has to perturb great deal of the obtained solution. This requires the attacker to develop a *new* optimization algorithm. For example, consider a design that has a total of 100000 gates. In the final solution S, 10000 nodes are visible (LUT or cell outputs) and therefore the average probability, that a node from the initial network is visible in the final solution, is $p = \frac{1}{10}$. If the watermarking strategy results in a pseudo-random selection of 1000 visible vertices, inherently, the average probability that a node, visible in S, is visible in a solution obtained by some other algorithm is p. That is, if the challenging algorithm retrieves a solution of the same quality. The probability expectation P, that some other algorithm selects exactly the same subset S of nodes in the final solution, is $P = p^{1000}$ or one in 10^{1000}. Consider that the attacker aims to reduce the likelihood of authorship by doing local changes to the design in order to remove the watermark. To reduce the proof of authorship to one in a million, the attacker has to alter 851 node from the watermark, i.e. 85.1% of the final solution. To remove the watermark in such a way that the remaining proof of authorship is $P = 0.1$, the attacker has to modify 888 vertices in the watermark or 88.8% of the entire technology mapping solution.

There are two scenarios how the attacker can try to find his or her signature in an already watermarked solution (see Figure 5). The first one is a top-down approach, where the attacker modifies the input hoping that the tool will produce an output that contains attacker's signature (as well as the author's signature). Since node permutation is pseudo-randomized, the likelihood that attacker's signature appears in the output is the same as the probability of two different algorithms retrieving the same solution. Thus, this attack is less efficient than trying to delete the signature.

In the bottom-up approach the attacker concludes from the output (or its modification), what is the input that produces output that contains her or his signature. However, in order to produce such an input (and possibly output), the attacker has to know which pseudo-random selection of nodes (and augmented network) corresponds to a specific input sequence. The attacker may obtain such information only if the reverse to the one-way function is known. For RSA-type one-way hash functions such inverses are not known [Men97].

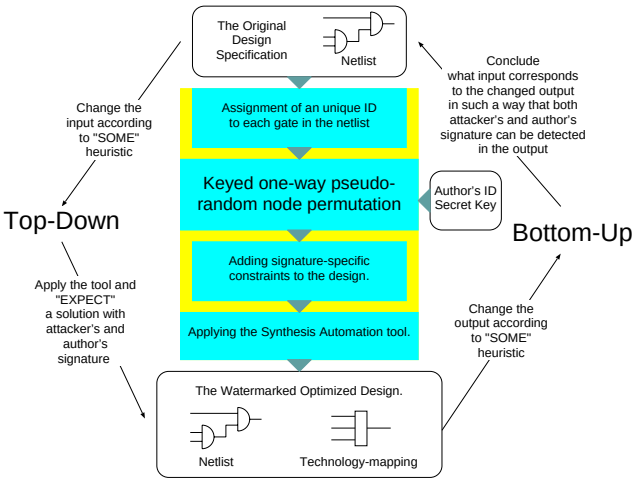

Figure 5: A top-down and bottom-up approach to finding attacker's signature in an already watermarked solution.

5. EXPERIMENTAL RESULTS

We demonstrate the effectiveness and quality of the developed IP protection approach on the problem of technology mapping for the set of MCNC benchmark designs (see Table 1). For LUT-based 5-input technology mapping we used the CutMap algorithm [Con96b]. Although the designs evaluated on the MCNC benchmark suite are much smaller than current industrial circuits (recently announced Xilinx Virtex series of FPGAs implemented using a 0.25 micron technology are expected to encompass 1,000,000 gates), we have achieved likelihood of watermarked solution coincidence on average equal to $p < 10^{-13}$ with average overhead of 4%. In two cases design watermarking resulted in negative overhead. Similarly, we obtained average $p < 10^{-26}$ with average hardware overhead of 7.6%.

We have applied the IPP protocol for technology mapping to a large industrial design example with over 47,000 non-primary

Circuit	PO	Non-PO	Orig	0.50%			1%			2%		
i7	67	439	139	139	0.00%	0.01891	141	1.44%	0.00040	111	-20.14%	1.69E-09
i2	1	530	121	121	0.00%	0.01952	123	1.65%	0.00042	127	4.96%	2.44E-07
i9	63	471	140	140	0.00%	0.01405	142	1.43%	0.00022	145	3.57%	7.05E-08
alu4	8	603	220	220	0.00%	0.04278	221	0.45%	0.00188	226	2.73%	4.69E-06
frg2	139	507	302	302	0.00%	0.05632	304	0.66%	0.00337	305	0.99%	1.21E-05
rot	107	593	287	287	0.00%	0.02916	287	0.00%	0.00085	291	1.39%	9.38E-07
apex6	99	628	242	242	0.00%	0.00960	244	0.83%	0.00010	249	2.89%	1.55E-08
C2670	140	716	330	330	0.00%	0.00866	331	0.30%	7.78E-05	335	1.52%	8.15E-09
x3	99	681	266	266	0.00%	0.00835	268	0.75%	7.55E-05	273	2.63%	8.49E-09
k2	45	820	446	448	0.45%	0.05434	449	0.67%	0.00301	453	1.57%	1.07E-05
i8	81	827	517	515	-0.39%	0.06952	505	-2.32%	0.00399	493	-4.64%	9.88E-06
dalu	16	1065	382	385	0.79%	0.00354	388	1.57%	1.36E-05	397	3.93%	3.10E-10
t481	1	1144	543	545	0.37%	0.01424	540	-0.55%	0.00018	545	0.37%	4.11E-08
C3540	22	1336	563	563	0.00%	0.00238	568	0.89%	6.43E-06	580	3.02%	7.39E-11
C5315	123	1373	460	459	-0.22%	6.36E-05	457	-0.65%	3.72E-09	474	3.04%	5.42E-17
pair	137	1426	520	525	0.96%	9.32E-05	535	2.88%	1.25E-08	554	6.54%	5.90E-16
C6288	32	2417	690	705	2.17%	1.95E-07	725	5.07%	7.70E-14	746	8.12%	2.51E-26
C7552	108	2441	764	774	1.31%	1.30E-07	788	3.14%	2.82E-14	809	5.89%	3.52E-27
des	245	2788	1141	1097	-3.86%	6.65E-08	1110	-2.72%	6.75E-15	1132	-0.79%	1.85E-28
i10	224	2974	1315	1324	0.68%	3.78E-07	1339	1.83%	2.13E-13	1356	3.12%	1.12E-25

Table 1. Watermarking technology mapping solutions for the MCNC suite. Columns present, respectively: name of the circuit, number of primary outputs, number of non-primary gates in the project description, and the solution quality (number of LUTs) when algorithm CutMap [Con96] is applied to the original design. Each three-column subtable contains a column describing the number of LUTs in the watermarked solution, the hardware overhead with respect to the non-watermarked solution, and the likelihood that some other algorithm retrieves a solution which also contains the watermark.

and 5,000 primary gates. For 0.5% and 1% of non-primary gates selected for assignment to pseudo-primary outputs, our approach resulted in solution coincidence likelihood of 10^{-124} and 10^{-244}, and with incurred hardware overhead of 0.8% and 1.87%, respectively. The run-time of the optimization program for the watermarked input was within $\pm 5\%$ of the program execution run-time for the original input.

The evaluation of the developed watermarking technique for multi-level logic minimization resulted in results similar to technology mapping. We applied the MIS suite of optimization algorithms [Bra87] to the standard and watermarked set of MCNC benchmark designs. After specifying 1% or 2% of non-primary output nodes to become pseudo-primary outputs, the MIS suite retrieved in average solutions with 2% fewer or 6% more literals, respectively.

6. CONCLUSION

We have developed the first watermarking-based approach for IPP of tools and designs in the combinational logic synthesis domain. The watermark, a set of constraints which correspond to the designer's and/or tool developer's signature, are added to the original design specification in a synthesis preprocessing step. After the synthesis tool retrieves a solution to the optimization problem, the added constraints are satisfied in addition to the original set of design constraints. This property is used to prove authorship in court. We demonstrated that the embedded watermarks are: hard to delete and hard to find in an arbitrary solution. We have effectively applied our approach to the problem of technology mapping for LUT-based FPGAs using a set of benchmark designs.

ACKNOWLEDGMENTS

Darko Kirovski was partially supported by NSF under grant CCB-9734166. Yean-Yow Hwang was partially supported by Quickturn under the California MICRO program.

7. REFERENCES

[Ben96] W. Bender et al. Techniques for data hiding. IBM Systems Journal, vol.35, no.3-4, pp.313-336, 1996.

[Ber96] H. Berghel and L. O'Gorman. Protecting ownership rights through digital watermarking. Computer, vol.29, no.7, pp.101-103, 1996.

[Bra87] R.K. Brayton et al. MIS: a multiple-level logic optimization system. TCAD, vol.6, no.6, pp.1062-81, 1987.

[Con96a] J. Cong and Y. Ding. Combinational Logic Synthesis for LUT Based Field Programmable Gate Arrays. Trans. on Design Automation of Electronic Systems, vol.1, no.2, pp.145-204, 1996.

[Con96b] J. Cong and Y.-Y. Hwang. Simultaneous Depth and Area Minimization in LUT-based FPGA Mapping. 3rd International Symposium on FPGA, pp. 68-74, 1995.

[Cox96] I.J. Cox et al. Secure spread spectrum watermarking for images, audio and video. Intl Conference on Image Processing, vol.3, pp. 243-246, 1996.

[DeM94] G. De Micheli. Synthesis and optimization of digital circuits. McGraw-Hill, New York, 1994.

[Hac96] G.D. Hachtel and F. Somenzi. Logic synthesis and verification algorithms. Kluwer, Boston, 1996.

[Har97] F. Hartung and B. Girod. Watermarking of MPEG-2 encoded video without decoding and re-encoding. Multimedia Computing and Networking, pp.264-274, 1997.

[Hon98] I. Hong and M. Potkonjak. IPP Techniques for Behavioral Specifications. Unpublished manuscript. 1998.

[Kah98] A.B. Kahng et al. Robust IP Watermarking Methodologies for Physical Design. Design Automation Conference, 1998.

[Lac98] J. Lach, W.H. Mangione-Smith, and M. Potkonjak. Fingerprinting Digital Circuits on Programmable Hardware. Workshop in Information Hiding, 1998.

[Lia97] S. Liao and S. Devadas. Solving Covering Problems using LPR-based Lower Bounds. Design Automation Conference, pp.117-120, 1997.

[Liu97] T.H. Liu et al. Optimizing Designs Containing Black Boxes. Design Automation Conference, pp.113-116, 1997.

[Men97] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. Handbook of applied cryptography. Boca Raton, 1997.

[Tiw96] V. Tiwari et al. Technology mapping for low power in logic synthesis. Integration. vol.20, (no.3), 1996.

[VSI97] VSI Alliance. Fall Worldwide Member Meeting: A Year Of Achievement. October 1997.