Protecting Combinational Logic Synthesis Solutions

Darko Kirovski, Yean-Yow Hwang, Miodrag Potkonjak, and Jason Cong

***Abstract*— Recently, design reuse has emerged as a dominant design and system integration paradigm for modern systems on silicon. However, the intellectual property business model is vulnerable to many dangerous obstructions, such as misappropriation and copyright fraud. We propose a new method for intellectual property protection which relies upon design watermarking at the combinational logic synthesis level. We introduce two protocols for embedding user- and tool-specific information into a logic network while performing multi-level logic minimization and technology mapping, two standard optimization processes during logic synthesis. The hidden information can be used to protect both the design and the synthesis tool. We demonstrate that the difficulty of erasing or finding a valid signature in the synthesized design can be made arbitrarily computationally difficult. In order to evaluate the developed watermarking method, we applied it to a standard set of real-life benchmarks, where high probability of authorship was achieved with negligible overhead on solution quality.**

I. Introduction

The ever-increasing complexity of designing modern systems-on-chip, as well as strong demands for shortened time-to-market, have resulted in design reuse as a predominant system development paradigm. The new core development strategies have affected the business model of virtually all VLSI CAD and semiconductor companies. For example, a number of companies have recently consolidated their efforts towards developing off-the-shelf programmable or application-specific cores (e.g., ARM, Tensilica, LSI Logic). In order to rapidly overcome the difficulties in core-based system design, the Virtual Socket Initiative 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.

In this manuscript we propose the first approach for intellectual property protection (IPP), which enables design watermarking while performing combinational logic synthesis. User- and/or potentially tool-specific copyright information of arbitrary length is embedded into the logic network of a design during a preprocessing step to traditional synthesis. The flow-graph for this process is shown in Figure 1. First, the copyright information is hashed using a cryptographically secure hash function such as SHA-256 [30] to create a key used to seed a cryptographically secure pseudo-random number generator. The resulting semi-infinite stream of pseudo-random bits is used to generate a unique set of design constraints which do not exist in the original specification. The constraints are conceived to be uniquely dependent upon the copyright information. By superimposing these constraints to the original logic network, a new input is generated which has the same functionality as the original specification and in addition, contains copyright-specific information. By applying an off-the-shelf synthesis tool to the watermarked input specification, we obtain a solution to both the original and constrained input. Proof of authorship is based on the fact that the likelihood of another application returning a solution to both the original and constrained input is exceptionally small. The developed technique is transparent to synthesis and can be used in synergy with any logic synthesis tool. It 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 with probability proportional to the cardinality of the augmented additional constraints. The most powerful mechanisms for protection of combinational logic synthesis realizations are not provided by the logic synthesis phase itself, but by the consequent design steps in the overall design flow, such as technology mapping, placement, and routing. In principle, and often in practice, it is possible to drastically alter combinational networks while preserving functionality. Therefore, potential attackers can sometimes remove the majority, if not all, of the embedded local watermarks. However, once the logic-level specification is altered, all consequent steps must be redone. Note that physical design phases most often dominate synthesis in terms of effort and time. In addition, all verification, validation, and simulation steps must be also conducted on the new specification. Hence, watermarking at the logic synthesis level is an attractive design option.
- **Protect the synthesis tool.** The signature of the tool developer, embedded in logic synthesis solutions, clearly indicates the origin of the synthesis tool.

The added constraints result in a synthesis trade-off. The more additional constraints, the stronger the proof of authorship. However, an increased number of additional constraints also implies increased expectation of a performance overhead for the final output specification. In this paper we demonstrate empirically, using a series of experiments conducted with standard and industry-strength benchmarks, that using our technology, strong proofs of authorship can be enabled at a marginal real-estate and performance overhead.

Darko Kirovski is with Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA, e-mail:darkok@microsoft.com. Yean-Yow Hwang is a member of the research and development staff at Velogix Inc., Santa Clara, CA 95054, USA. Miodrag Potkonjak and Jason Cong are with the Computer Science Department, University of California, Los Angeles, CA 90095, USA.

Yean-Yow Hwang was partially supported by Quickturn under the California MICRO program.

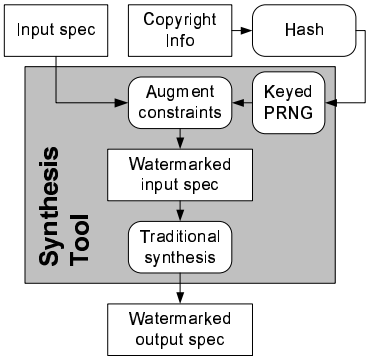


Fig. 1. The generic paradigm for watermarking solutions to combinatorial optimization problems.

The developed protocols for watermarking solutions of combinational logic synthesis exactly follow the requirements identified in the Strawman proposal of the Development Working Group on IPP. The recognized desiderata reflects:

- **Functionality Preservation.** Functional requirements should not be altered by the application of IPP tools.
- **Minimal Hassle.** The technique should be transparent to the already complex design and verification process.
- **Minimal Cost.** Both the cost of applying the protection technique and its hardware overhead should be low.
- **Enforceability.** The technique should provide strong and undeniable proof of authorship.
- **Flexibility.** It should enable a spectrum of protection levels which correspond to variable cost overheads.
- **Relative Persistence.** The removal of the embedded watermark should result in a task of the difficulty equal to the complete redesign of the specified functionality.

Figure 1 shows the steps required for watermarking-based IPP of solutions to combinatorial optimization problems. The encrypted signature of the author is translated into additional constraints to the formulation of the problem and the problem is solved using standard techniques and tools. A preliminary version of this work was published in [24].

A. Layered Protection of Intellectual Property

The problem of effective IPP in the competitive EDA environment has been addressed at the level of physical design [21] and behavioral specification [18] (see related work in Section II). Achieved protection of logic synthesis tools and designs, represented at levels of abstraction equal to or lower than netlists, is an obvious contribution of the work presented here. In addition, we must argue about the effectiveness and need for embedding signatures at the logic synthesis level for overall design protection. First, watermarking behavioral specification often does not exhibit sufficient potential for embedding large signatures which are crucial for high authorship credibility. Then, physical layout should not be an exclusive domain for IPP, due to the following two reasons:

- Modern *reverse engineering* technologies enable inexpensive, precise, fast, and most importantly, confidential retrieval of the netlist of a silicon product.
- *Firm cores* (sold in a netlist format) are of particular interest to system integrators because they more efficiently explore the physical design trade-offs.

Protection of solutions to logic synthesis is a crucial link in the full copyright protection system. Watermarks from each synthesis step can be removed by reverse engineering of the source design and resynthesizing the reverse engineered spec. They can also be removed by aggressive peephole optimization which must alter a substantial part of the design [23]. Thus, a watermark embedded only at the lowest design level is not as secure as a series of watermarks embedded at each design step. By embedding watermarks in a layered fashion, i.e., at each design stage, the adversary is forced into reverse engineering of the low-level design all the way up to the highest behavioral specification. This effort then must be followed by a design starting from this top-most spec – an effort comparable to designing the entire system from scratch.

B. Motivational Example

We introduce the IPP approach for logic synthesis solutions and the intuition behind the data hiding protocols by embedding a copyright-specific signature into the solution of a simple library binding example. Consider a six-input single-output logic network depicted in Figure 2(a). The network consists of eleven gates. The goal of the optimization approach is to map the network to as few as possible cells from the library given in Figure 2(c). Finding an area-optimal solution to the technology mapping problem is NP-hard [15]. In this simple example, the optimum cover uses six cells. In addition, the cardinality $|S|$ of the set S of all possible mapping solutions is $|S| = 49$. We obtained this result by performing exhaustive solution enumeration. Assuming that the probability of a particular mapping containment in a solution is uniform, the likelihood that a specific optimization algorithm returns exactly the same solution as another corresponds to $p = \frac{1}{|S|} = \frac{1}{49}$.

The goal of the watermarking approach is to embed additional constraints, which uniquely correspond to the author's signature, into the problem specification, such that the final solution can be retrieved only within a subset S_{sub} of the set of all solutions S. In that case, the proof of authorship will be as strong as the probability p that a random solution is retrieved from S_{sub}, i.e., $p = \frac{|S_{sub}|}{|S|}$.

Consider the following simple protocol for embedding constraints. As shown in Figure 2(a), the nodes (gates) of the network are uniquely identified. We denote the set of all node identifiers as N. In the network in Figure 2(a), $N = \{1, 2, \ldots, 10\}$. The user-specific data, supposed to be augmented into the solution, is a subset $N_o \subset N$, where the cardinality $|N_o|$ is typically smaller than $|N|$. We impose constraints to the problem specification by assigning nodes (the outputs of gates) that have identifiers equivalent to the numbers in the selected subset N_o, to become pseudo-primary outputs in the input design specification. This action imposes that the restricted gates have outputs visible in the final library binding. The resulting binding is obtained by feeding the augmented design specification as input to the synthesis tool. The pseudo-primary outputs are visible in the final solution

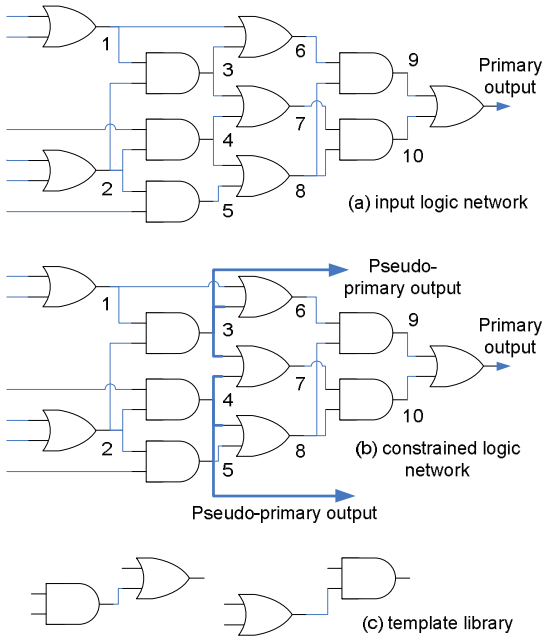


Fig. 2. An example of watermarking technology mapping solutions. Subfigure (a) illustrates a logic network. Subfigure (b) illustrates the input additionally constrained by setting points 3 and 4 as primary output and solution to template matching. Subfigure (c) depicts the template library.

regardless of the applied library binding algorithm. Finally, the length n of the user-specific bit-stream that identifies N_o must be greater than or equal to:

$$n \geq \log_2 \binom{|N_o|}{|N|}. \quad (1)$$

If we assume that $|N_o| = 2$ and $|N| = 10$, we obtain minimal message length of $\log_2 45 \leq 6$ bits. Note that the cardinality of the space of all possible solutions, including the ones with inferior covers, is higher than 45 (e.g., there are $|S| = 49$ optimal solutions). However, the hope of the designer is that the synthesis tool still produces an optimal output. For example, assume that the encoded message is $\{3, 4\}$. By following the afore-described protocol, we obtain a constrained logic network shown in Figure 2(b). The constrained network can be still solved using six standard cells. However, there exists a set $S_{3,4}$ of only four solutions of minimal cell cardinality. Therefore, an algorithm applied to this restricted network can retrieve one of these four solutions. Assuming an optimal solution has been found, the probability that another algorithm applied to the initial problem specification returns a solution from $S_{3,4}$ is equal to $p = \frac{|S_{3,4}|}{|S|} = \frac{4}{49}$. This probability demonstrates the strength of the proof of authorship. This example is rather small, resulting in a small solution space. As shown in the experiments, real-life examples have a much larger potential for embedding data with little or no loss of solution quality. Then, the achieved strength of authorship is well beyond accidental coincidence.

In general, in order to evaluate the efficacy of any IPP scheme, the following questions have to be answered. How strong is the proof of authorship with respect to the amount of hidden information? How much overhead in the solution quality is incurred by a watermark of particular cardinality? How easy is it to remove the IPP? According to the applied IPP protocol, how easy is it to find someone else's signature in a particular solution? In this paper we will answer these questions for the developed watermarking protocols for multi-level logic minimization and technology mapping.

II. Related Work

A. Functional Artifact Watermarking

In watermarking the functional artifacts, several methods have been proposed since the original submission of this paper. The watermarking-based hardware IPP technique can be classified according to the level of abstraction at which a particular design spec is watermarked and according to the employed watermarking protocol. Watermarking techniques have been proposed at all levels of the design process, including the algorithm- [18], [33], [5], [25] and system-level [23], logic synthesis [29], [32], FPGA-based logic synthesis [26], [27], and physical synthesis [6], [7], [21], [22], [31]. In addition to watermarking digital designs, several efforts were dedicated to protect analog and mixed signal designs [20].

B. Combinational Synthesis

Combinational logic synthesis has been thoroughly studied. A detailed description of optimization problems and a good survey of minimization techniques and existing non-commercial synthesis frameworks are presented in [15], [17]. The targets of the latest improvements in combinational logic synthesis are refined covering algorithms [28], new applications of decision diagrams [2], etc. Similarly, the research activity in technology mapping has been streamlined towards library- or LUT-targeted [8], [12], [13], [14] algorithms.

Our approach is the first to define a set of protocols for information hiding into a design at the logic synthesis level. Such a watermarking technique provides security against sophisticated reverse engineering attacks, enables concurrent physical layout optimization of a purchased off-the-shelf core and its protection, and exhibits greater potential for adding larger amounts of information than IPP techniques at the behavioral synthesis level.

III. Preliminaries

Here we briefly outline the definitions of problems solved in multi-level logic minimization and technology mapping.

A. Multi-level Logic Minimization

Many technology parameters drive the logic minimization process towards multi-level logic networks. Such networks, besides obeying the technology requirements, enable the synthesis tool to exploit various minimization trade-offs. Common optimization targets at this step are: area, critical path, power consumption, testability, etc. The input to a multi-level minimization process is a set of Boolean onset and associated don't-care functions applied to a set of variables. With no loss of generality, the input to the multi-level minimization is represented using a logic network. The exact definition of a

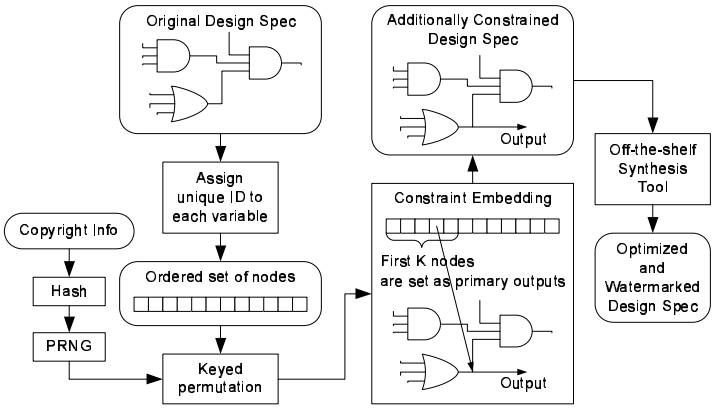


Fig. 3. The protocol for hiding information in solutions to multi-level logic optimization and technology mapping.

non-hierarchical combinational logic network is given in ([15], Def.8.2.1). Given a logic multi-level network described using gates exclusively from a given library, the goal of a multi-level logic minimization strategy is to find an equivalent alternate multi-level representation of the initial network for which the set of targeted design properties is optimal. A good survey of optimization strategies for multi-level logic minimization is given in [15], [17].

B. Technology Mapping (or Cell-Library Binding)

In this step of logic synthesis, the output of the multi-level logic minimization is mapped to a predefined cell library for standard cell-based ASIC designs or a network of Look-Up Tables (LUTs) for FPGA designs. The mapping is performed in such a way that the original network is partitioned into subnetworks, which are functionally equivalent to the set of cells exclusively selected from a given cell library. The tool developer and the user are concerned about the logic abstraction of the cell library and associated implementation parameters such as area, delay, etc. The mapping is performed with respect to a set of timing, area, and/or power consumption constraints and/or minimization goals. Two large classes of algorithms exist that target this set of problems. The first one is the Boolean class where the library cells and the portion of network of interest are represented using Boolean functions. A survey of approaches from this class of algorithms is given in [2], [10]. The second class of algorithms uses the structural properties of the input network. Such an algorithm facilitates algebraic decompositions of the input logic network to generate a new network of smaller gates (gate decomposition) and then covers it with cells or LUTs (LUT covering). The result is functionally equivalent to the original. A survey of techniques of this class is presented in [9], [15], [17].

IV. IPP of Combinational Logic Synthesis

We developed an IPP strategy that enables the designer or tool developer to embed a watermark into the optimized design at each level of combinational logic synthesis. The key idea is to augment information into the initial specification of the design in such a way that after one of the combinational synthesis steps is applied, the resulting design is both functionally correct and, in addition contains a probabilistic proof that it has been created by the designer and/or the applied tool.

A. The Synthesis Flow

The synthesis flow for the IPP of combinational logic synthesis solutions is shown in Figure 3. The first three phases in the watermarking approach are the same for both multi-level logic minimization and technology mapping. In the first step, the gates in the initial logic network specification are sorted using an industry specified standard. As a result of this procedure, each gate is assigned a unique identifier. Next, the gate ordering is permuted in a way specific to the copyright information. The length of the copyright information is arbitrary. It is hashed using a cryptographically secure hash function such as SHA-256 [30] to create a pseudo-uniquely corresponding key of fixed length (256 bits). This key is then used to seed a cryptographically secure random number generator (PRNG) such as the keyed RC4-based PRNG [30]. The resulting copyright-specific semi-infinite bit-stream guides the process of permuting all uniquely identified design variables.

In the next phase, the first K variables in the pseudo-random permutation are selected for explicit assignment to primary outputs. In the case of technology mapping, this phase represents the final phase in the watermarking protocol. If multi-level logic minimization is performed, the generated pseudo-primary outputs are used as inputs into an additional logic network which is embedded into the initial design specification. This network is created according to the generated copyright-specific bit-stream. 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 likelihood of accidental watermark existence in a netlist or technology-mapping determines the proof of authorship. The strength of the proof is proportional to the cardinality of the augmented constraints. The proposed IPP scheme is developed to satisfy the identified requirements for secure and efficient design watermarking. In the remainder of this section, we present the details of the developed IPP protocols, discuss their security and privacy properties, and finally, dis-

cuss whether the protocols meet the identified desiderata for efficacy of an information hiding scheme.

B. Gate Ordering

The objective of the initial phase is to assign a unique identification number (ID) to each node in the input logic network. This assignment aims to indisputably correlate the embedded data with the input logic network. To disable misinterpretation of this ordering, an industry standard has to be established. The network has to be numbered in such a way that only identical nodes, i.e., nodes with the same functionality and isomorphic and functionally equivalent transitive fanin and fanout graphs, are not assigned unique IDs. The problem of finding whether two nodes satisfy these constraints is a hard problem. Its special case, 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 [1], [16]. Thus, we do not attempt to solve the generic superclass of the graph isomorphism problem. Instead, we propose a mapping function that exploits the functional and timing properties of each node, as well as a set of restriction rules, in order to assign a unique ID to each gate.

The gate ordering algorithm is described in detail using the pseudo-code in Figure 4. Initially, all gates in the logic network that are not primary outputs constitute the starting set of nodes M_0. The goal of the gate ordering algorithm is to partition M_0 into an ordered set of single-gate partitions using a set of ordering rules $Cl, l = 1 \dots 8$. We denote the current set of partitions as $\mathcal{M}$. Thus, initially $\mathcal{M} = \{M_0\}$. The algorithm iteratively partitions the current set of gates M_i into subsets such that all gates in each subset have the same value for the current ordering criterion Cl. If the current set M_i is successfully partitioned, then the new partitions are appended to $\mathcal{M}$ and M_i is removed from $\mathcal{M}$. The next iteration aims at partitioning the new M_i. If M_i cannot be partitioned, then i is incremented. This process is iterated until all subsets are processed. If two nodes cannot be distinguished using the set of rules, i.e., they are assigned the same ID, we exclude them from the set of nodes used to embed the watermark. Thus, nodes that cannot be uniquely identified are not considered in the IPP process. Note that such nodes are not likely to occur in logic designs due to typically asymmetric variable dependencies.

Definition 1: **Input/Output Node Level.** A node G_i has an input/output level K if the longest path in the logic network from any input/output to G_i is of length K.

We denote the input and output level of a node G_i as λ_i^{IN} and λ_i^{OUT} respectively. We propose the following list of eight criteria for node identification:

- **C1** The input level of node G_i.
- **C2** The output level of node G_i.
- **C3** This is a set of criteria that describes the subgraph that computes the inputs to node G_i. Criterion **C3[j]** considers the number of nodes in the transitive fanin of G_i at level $j < \lambda_i^{IN}$. The cardinality of **C3** for a node G_i at level λ_i^{IN} is $\lambda_i^{IN} - 1$.
- **C4** This is a set of criteria that describes the subgraph that uses the outputs of node G_i. Criterion **C4[j]** considers the number of nodes in the transitive fanout of G_i at level $j < \lambda_i^{OUT}$. The cardinality of **C4** for a node G_i at level λ_i^{OUT} is $\lambda_i^{OUT} - 1$.

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

Input = ordered set $\mathcal{M}$ of sets of nodes M,
where initially $\mathcal{M} = \{M_1 = L - O\}$ has one element.
for each Criterion Function $Cl, l = 1 \dots 8$
  for each set of nodes $M_i \in \mathcal{M}$ with $|M_i| > 1$
    Set $x = |\mathcal{M}|$.
    for each node $G_j \in M_i$
      Compute $\alpha(l, j) = Cl(G_j)$.
    Partition M_i into an ordered set of K unordered sets $\{M_{x+1}, \dots, M_{x+K}\}$ such that $(\forall G_j \in M_{x+k}, 1 \le k \le K)\ \alpha(l, j) = \text{const.}$ and $(\forall G_j \in M_{x+k}, \forall G_m \in M_{x+k+1})\ \alpha(l, j) > \alpha(l, m)$ for any $1 \le k \le K - 1$.
    if M_i has been partitioned
      $\mathcal{M} = \mathcal{M} \cup \{M_{x+1}, \dots, M_{x+K}\}$
      Remove M_i from $\mathcal{M}$.
    else i++
for each set of nodes $M_i \in \mathcal{M}$
  if $|M_i| > 1$
    Remove M_i from $\mathcal{M}$.

Fig. 4. Pseudo-code of the proposed function for gate ordering.

The following set of criteria considers the functionality and fanin and fanout factors of the nodes that belong to the transitive fanin and fanout subgraphs for node G_i. The established EDA standard should assign a unique identifier to each type of gate (e.g., AND gate is assigned 0, OR 1, etc.). When considering a group of nodes, the ID is built by concatenating a sorted list of nodes' gate identifiers. Similarly, the fanin and fanout factors represent the number of inputs and the number of gates that use the output of a particular gate, respectively. When considered in a group, the factors are concatenated in a sorted list. We propose the following four sets of criteria for a given node:

- **C5** This is a set of criteria that describes the subgraph that computes the inputs to node G_i. Criterion **C5[j]** considers the functionality, fanin, and fanout of nodes in the transitive fanin of G_i at level $j < \lambda_i^{IN}$. The cardinality of **C5** for a node G_i at level λ_i^{IN} equals $3F_i^{IN}$, where F_i^{IN} is the cardinality of the transitive fanin of G_i.
- **C6** This is a set of criteria that describes the subgraph that uses the outputs of node G_i. Criterion **C6[j]** considers the functionality, fanin, and fanout of nodes in the transitive fanout of G_i at level $j < \lambda_i^{OUT}$. The cardinality of **C6** for a node G_i at level λ_i^{OUT} equals $3F_i^{OUT}$, where F_i^{OUT} is the cardinality of the transitive fanout of G_i.

If two nodes cannot be distinguished using the above six criteria, we recursively apply the last two criteria to the nodes in the transitive fanin and fanout of node G_i.

- **C7** Recursive application of **C5** and **C6** to all nodes in the transitive fanin of G_i sorted by level, then, if more nodes exist per level, sorted by value.
- **C8** Recursive application of **C5** and **C6** to all nodes in the transitive fanout of G_i sorted by level, then, if more nodes exist per level, sorted by value.

C. Watermark Encoding and Embedding

In the next step of the IPP process, we perturb the node ordering in a keyed one-way fashion with respect to the copyright data. This action is taken in order to prevent alteration of the input (i.e., copyright information) in order to achieve a desired output (i.e., copyright-specific constraints). The attacker may change heuristically the input, observe the output, and conclude which action should be taken in order to remove, modify, or alter the existing watermark.

A myriad of implementations exists for this preventive step. We use a sequence of cryptographically secure hashing and pseudo-random number generation [30] in order to generate a sequence of bits that decides upon the node selection. The result of this phase in the protocol is a copyright-specific permutation of network nodes. The first K nodes in the resulting permutation are selected and marked as pseudo-primary output as shown in Figure 3. By performing this step, the watermarking routine enforces these nodes to be:

- **visible** in the final technology mapping solution, and
- **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.

In the case of **technology mapping**, the encoded constraints are made visible in the final mapping by explicitly specifying their outputs as primary (pseudo-primary). However, it is important to stress the implications of a specific phenomenon in this problem. Cong and Ding have identified that input and output nodes of maximum fanout-free cones in the logic network (i.e., MFFC nodes), are more likely to appear in the final solution than the remaining nodes [9]. We 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 enumerated explicitly the ratio of MFFC nodes in the initial input specification (rin) and in the final solution ($rout$). Then, we computed the likelihood of solution coincidence using the following formula:

$$p = \left[\frac{rout \cdot F}{rin \cdot T}\right]^{rout \cdot K} \cdot \left[\frac{(1 - rout) \cdot F}{(1 - rin) \cdot T}\right]^{(1-rout) \cdot K}, \quad (2)$$

where F is the number of internal gates in the final solution, T is the total number of internal gates in the initial logic network, and K 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 in multi-level logic minimization. However, we propose an alternative protocol that provides a stronger proof of authorship due to additional embedded constraints.

In the case of **multi-level synthesis**, the watermark is embedded in two phases. In the first phase, the protocol marks the outputs of selected gates as visible by explicitly denoting them as pseudo-primary. In the second phase, an additional netlist is augmented into the original netlist. The additional netlist has as input variables the pseudo-primary output variables generated in the previous phase. This netlist is also built using the semi-infinite bit-stream pseudo-randomly produced using the copyright information. The pseudo-code for building the augmented netlist is presented in Figure 5.

The netlist generation is done by iterating the following procedure. First, a gate G from the available GATELIBRARY is randomly selected. All selections during the algorithm are guided by the copyright-specific pseudo-random bit-stream. Then, we randomly select $G.fanin$ pseudo-primary outputs from O and use them as input to G. The output of G is added to the list of pseudo-primary outputs O. This output is subject to selection in future iterations of the procedure. Gate appending can be repeated as desired. A possible termination policy may be established using adopted standards by the EDA industry. The standards may enable customizing the influence of the proof of authorship on the induced overheads. Namely, the application of this procedure may result in significant cumulative run-time and typically in marginal performance or real-estate overhead.

We denote as L the input logic network,
as M_o the ordered set of pseudo-primary outputs,
and as r the pseudo-random bit-stream.
Repeat according to an EDA Standard
  Gate G = select from GATELIBRARY based upon r.
  Select $G.fanin$ outputs from M_o based upon r
  and use them as input to G.
  $M_o = M_o \cup \{G\}$.
End Repeat

Fig. 5. Pseudo-code of the proposed function for watermarking multi-level logic minimization solutions using network augmentation.

The additionally constrained netlist is now fed to an off-the-shelf (technology mapping or multi-level logic minimization) synthesis algorithm, which produces the output design spec. In the case of technology mapping, the solution is a network of templates with the set of copyright-determined pseudo-primary outputs as a subset to the set of all nodes in the template network. Proof of authorship is based upon the fact that the likelihood is relatively low for all or most copyright-specific pseudo-primary outputs to be "visible" in a solution produced by an off-the-shelf optimizer not aware of the copyright-specific constraints.

In the case of multi-level logic minimization, the solution is conceived with the influence of the copyright-specific augmented logic sub-network. Note that this sub-network does not exist in the final design specification; it is removed after the synthesis step. However, its effect on subfunction selection in the functionally important part of the design specification is significant. Proof of authorship relies on the fact that it is unlikely for an off-the-shelf synthesis tool to produce the same set of subfunctions in the optimized netlist with access to the original design spec only and without knowledge of the augmented copyright-specific logic network. In both cases, to complicate the adversary's position, due to the fixed gate ordering and the application of the one-way pseudo-random permutation, the adversary cannot conclude how the

watermark changes based upon particular input modification. Thus, the adversary cannot expect to adjust the input spec, apply an arbitrary optimization tool, and obtain output that contains a desired copyright information with high probability. Finally, the most important property of the proposed protocols is that they embed signatures transparently upon the used synthesis tools.

D. Watermark Detection

The embedded watermark is detected within an existing logic network using a simple procedure that can possibly incorporate exhaustive search in the case of high-effort attacks. Given a logic network, the detector computes the function of each node with respect to the primary inputs and compares it to the list of functions of nodes in the suspected original design. This establishes the node identifiers for the design under test. Note that under attack, certain nodes may not appear in the pirated design. We assume that in the case of a low-effort attack (several local peephole optimizations and/or added circuitry), many of the nodes are retained in the pirated design. The effect of node disappearance on the proof of authorship can be evaluated from Eqn.2. From Table I, we can observe that even designs with as few as several tens of pseudo-primary outputs can produce sufficiently strong proofs of authorship. By altering larger portions of the design, the adversary will clearly, at some point, obfuscate the watermark beyond the detection ability of a specific detector. For large designs, where several thousand pseudo-primary outputs can be easily created during IPP with marginal effect on circuit performance, the fact that only several hundred of them are required for positive detection points to the fact that in order to remove the watermark, the adversary must undertake a task similar to the redesign of more than 90% of the circuit.

Here, we address a specific issue in watermark detection: the case where a given netlist is embedded in another design. We assume that the detector has access to an isolated pirated design. Performing the design extraction from an incorporating design corresponds to the sub-graph isomorphism problem which remains open in terms of its complexity [1], [16]. In particular, if some parts of the design are obfuscated, this task becomes even more complex. One way to address this problem is by analyzing relatively small fanin and fanout subgraphs for each node in the netlist and comparing them to the corresponding subgraphs of the pseudo-primary outputs in the protected design. Matching these subgraphs may aid the problem of design extraction. Due to the complexity of these procedures, the description of tools that could perform extraction of pirated designs is beyond the scope of this manuscript. However, this remains a topic for future work.

E. Resistance to Possible Attacks

In the remainder of the section, we discuss the most effective attack scenarios on the proposed protocol.

1) Watermark Removal: In this first scenario, the adversary may try to modify the output locally in such a way that the watermark disappears or the proof of authorship is lowered below a predetermined standard. For example, peephole optimization (PO) by its nature, is a localized alteration of a given design. The question posed is whether it can affect an embedded watermark beyond the locality of its application. Here, we argue that if PO is applied at few places in the design, it will affect only a small part of the watermark. Similarly, global application of PO should remove a large part of the watermark. Let's assume PO is applied in a small locality of the design. In that case, such a change can indeed alter the gate ordering in a large part of the solution. To prevent this, the detector matches all nodes in the "suspect" design with all nodes in the "original" design as described in Subsection IV-D. From the assumption that a large part of the design is not affected by PO, this objective should be a relatively simple task. Once the matching is done, the detector proceeds with the constraint validation. Since a large part of the design has not been altered, the detector should indeed identify most of the author-specific constraints and authenticate the design.

Therefore, the watermarking scheme has to be such that to delete the watermark and still preserve solution quality, the adversary has to perturb a great deal of the obtained solution. This leaves the adversary with a need to develop a *new* optimization algorithm or a *new* design specification.

For example, consider a design that has a total of 10^5 gates. In the final solution S, 10^4 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 true, 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 adversary's aim is to reduce the likelihood of authorship by making local changes to the design in order to remove the watermark. To reduce the proof of authorship to one in a million, the adversary has to alter 851 nodes in 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 adversary has to modify 888 vertices in the watermark or 88.8% of the entire solution.

Finally, note that permutation of input or output pins cannot affect the watermark detection because netlist nodes are enumerated regardless of pins identifiers (see Figure 4). By permuting any of the pins, gate ordering does not change.

2) The adversary finds his watermark in a watermarked solution: Consider the case where the adversary wants to misappropriate the synthesis tool. The adversary has to modify the input in such a way that the tool will produce an output that contains desired copyright data. Note that this has to happen for each processed instance of the problem. Since the node permutation is pseudo-random, the likelihood that

Circuit	#PO	#IP	#LUT	0.50%			1%			2%			4%		
i7	67	439	139	139	0.00%	0.019	141	1.44%	0.0004	111	-20.14%	1.69E-09	143	2.88%	4.22E-14
i2	1	530	121	121	0.00%	0.019	123	1.65%	0.0004	127	4.96%	2.44E-07	134	10.74%	1.87E-13
i9	63	471	140	140	0.00%	0.014	142	1.43%	0.0002	145	3.57%	7.05E-08	152	8.57%	2.33E-14
alu4	8	603	220	220	0.00%	0.043	221	0.45%	0.0019	226	2.73%	4.69E-06	235	6.82%	5.84E-11
frg2	139	507	302	302	0.00%	0.056	304	0.66%	0.0034	305	0.99%	1.21E-05	311	2.98%	3.01E-10
rot	107	593	287	287	0.00%	0.029	287	0.00%	0.0008	291	1.39%	9.38E-07	300	4.53%	2.73E-12
apex6	99	628	242	242	0.00%	0.0096	244	0.83%	0.0001	249	2.89%	1.55E-08	255	5.37%	6.41E-16
C2670	140	716	330	330	0.00%	0.0087	331	0.30%	7.78E-05	335	1.52%	8.15E-09	354	7.27%	9.51E-16
x3	99	681	266	266	0.00%	0.0083	268	0.75%	7.55E-05	273	2.63%	8.49E-09	287	7.89%	5.93E-16
k2	45	820	446	448	0.45%	0.0543	449	0.67%	0.0030	453	1.57%	1.07E-05	459	2.91%	1.84E-10
i8	81	827	517	515	-0.39%	0.0695	505	-2.32%	0.0040	493	-4.64%	9.88E-06	494	-4.45%	1.06E-10
dalu	16	1065	382	385	0.79%	0.0035	388	1.57%	1.36E-05	397	3.93%	3.10E-10	398	4.19%	1.07E-19
t481	1	1144	543	545	0.37%	0.014	540	-0.55%	0.0002	545	0.37%	4.11E-08	546	0.55%	1.84E-15
C3540	22	1336	563	563	0.00%	0.0024	568	0.89%	6.43E-06	580	3.02%	7.39E-11	589	4.62%	1.28E-20
C5315	123	1373	460	459	-0.22%	6.36E-05	457	-0.65%	3.72E-09	474	3.04%	5.42E-17	489	6.30%	2.92E-32
pair	137	1426	520	525	0.96%	9.32E-05	535	2.88%	1.25E-08	554	6.54%	5.90E-16	555	6.73%	3.99E-31
C6288	32	2417	690	705	2.17%	1.95E-07	725	5.07%	7.70E-14	746	8.12%	2.51E-26	764	10.72%	7.01E-51
C7552	108	2441	764	774	1.31%	1.30E-07	788	3.14%	2.82E-14	809	5.89%	3.52E-27	827	8.25%	1.48E-52
des	245	2788	1141	1097	-3.86%	6.65E-08	1110	-2.72%	6.75E-15	1132	-0.79%	1.85E-28	1150	0.79%	3.21E-55
i10	224	2974	1315	1324	0.68%	3.78E-07	1339	1.83%	2.13E-13	1356	3.12%	1.12E-25	1371	4.26%	5.98E-50

Circuit	#PO	#IP	#LUT	8%			12%			16%		
i7	67	439	139	141	1.44%	6.98E-28	152	9.35%	2.73E-38	171	23.02%	3.64E-39
i2	1	530	121	153	26.45%	1.00E-23	163	34.71%	1.82E-33	185	52.89%	8.09E-35
i9	63	471	140	171	22.14%	7.94E-25	184	31.43%	4.36E-34	201	43.57%	6.99E-36
alu4	8	603	220	246	11.82%	3.34E-20	256	16.36%	1.20E-28	276	25.45%	1.86E-30
frg2	139	507	302	328	8.61%	4.15E-18	337	11.59%	1.43E-25	351	16.23%	1.32E-27
rot	107	593	287	311	8.36%	1.04E-22	324	12.89%	8.55E-32	341	18.82%	2.97E-34
apex6	99	628	242	268	10.74%	2.29E-29	287	18.60%	3.36E-40	311	28.51%	3.43E-42
C2670	140	716	330	371	12.42%	7.22E-29	390	18.18%	5.46E-40	415	25.76%	2.20E-42
x3	99	681	266	305	14.66%	5.12E-29	318	19.55%	5.45E-41	337	26.69%	2.96E-44
k2	45	820	446	473	6.05%	3.00E-19	491	10.09%	9.45E-27	506	13.45%	1.94E-29
i8	81	827	517	461	-10.83%	4.53E-23	503	-2.71%	1.01E-29	497	-3.87%	2.82E-35
dalu	16	1065	382	425	11.26%	3.88E-36	450	17.80%	1.50E-50	475	24.35%	3.15E-55
t481	1	1144	543	563	3.68%	5.61E-29	575	5.89%	7.63E-42	590	8.66%	6.67E-47
C3540	22	1336	563	613	8.88%	1.38E-38	641	13.85%	2.72E-54	673	19.54%	3.99E-59
C5315	123	1373	460	527	14.57%	4.40E-59	569	23.70%	3.48E-81	622	35.22%	3.21E-85
pair	137	1426	520	596	14.62%	6.88E-57	631	21.35%	1.65E-79	680	30.77%	1.94E-84
C6288	32	2417	690	857	24.20%	5.43E-91	934	35.36%	6.96E-125	1001	45.07%	4.78E-135
C7552	108	2441	764	901	17.93%	4.42E-96	951	24.48%	5.59E-136	1019	33.38%	5.23E-147
des	245	2788	1141	1225	7.36%	5.31E-102	1299	13.85%	4.62E-142	1365	19.63%	2.54E-155
i10	224	2974	1315	1429	8.67%	4.48E-94	1489	13.23%	3.22E-133	1552	18.02%	1.64E-146

TABLE I

EXPERIMENTAL RESULTS: WATERMARKING LUT-BASED TECHNOLOGY MAPPING SOLUTIONS FOR THE MCNC BENCHMARK SUITE. COLUMNS INDICATE RESPECTIVELY: NAME OF THE CIRCUIT, NUMBER OF PRIMARY OUTPUTS (#PO), NUMBER OF INTERNAL GATES IN THE PROJECT DESCRIPTION (#IP), AND THE SOLUTION QUALITY (#LUT) WHEN ALGORITHM CUTMAP [8] IS APPLIED TO THE ORIGINAL DESIGN SPECIFICATION. NEXT, THERE ARE SEVEN SUBTABLES FOR DIFFERENT PERCENTAGES OF NODES (GATES) BEING CONSTRAINED AS PSEUDO-PRIMARY OUTPUTS. 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 THE WATERMARK IS NON-INTENTIONAL.

the desired copyright accidentally appears in the output is the same as the probability of two different algorithms produce the same solution. Thus, this attack is less efficient than trying to delete the signature. In the case where the adversary wants to misappropriate a particular solution, he has to solve the following problem: given a solution, obtain an input such that its watermarking corresponds to a particular node selection. For this purpose, the adversary has to "break" the one-way hash function, i.e., find its reverse. Then the adversary has a chance to obtain a representative input by modifying the output. Note that the freedom to modify the input is limited by the standardized node ordering of the input network. Such ordering restricts the adversary from having any flexibility to generate different heuristic starting points after the randomized permutation step.

V. EXPERIMENTAL RESULTS

We demonstrate the effectiveness and quality of the developed IPP techniques on the problem of technology mapping for the set of MCNC benchmark and six industrial strength designs. We used CutMap as a state-of-the-art algorithm for LUT-based technology mapping [11].

Tables I and II show the results obtained when the algorithm was applied to an original and additionally constrained (watermarked) design from the MCNC benchmark suite and from a set of six available industry-strength designs. The first four columns in both tables specify the name of the mapped circuit, the number of primary outputs in the design specification, the number of gates with outputs that are not primary, and the number of LUTs required to map the design when CutMap is applied to the original design. The remaining groups of three columns quantify the performance overhead and security of watermarking when some specified percentage of internal nodes in the original design specification is marked as pseudo-primary (see Section IV). Within each three-column grouping, the first column presents the number of LUTs in the final solution, the second quantifies the overhead with respect to the non-watermarked solution, and the third column describes

circuit	PO	IP	#LUT	0.50%			1%		
a.flat	618	1045	947	947	0.00%	0.00238488	951	0.42%	6.45E-06
b.flat	8010	20238	12091	12177	0.71%	3.54E-70	12245	1.27%	3.31E-138
c.flat	3934	26574	9825	9942	1.19%	1.59E-86	10017	1.95%	6.86E-171
d.flat	7452	25727	14996	15080	0.56%	1.21E-68	15175	1.19%	3.55E-135
e	5142	47470	19338	19492	0.80%	4.79E-124	19700	1.87%	2.13E-244
f	6832	86058	41721	41991	0.65%	5.28E-168	42172	1.08%	3.87E-333

circuit	2%			3%			4%		
a.flat	959	1.27%	6.84E-11	966	2.01%	1.07E-15	968	2.22%	1.39E-20
b.flat	12424	2.75%	2.08E-268	12593	4.15%	2.98E-393	12768	5.60%	2.2E-509
c.flat	10237	4.19%	1.48E-330	10462	6.48%	1.22E-486	10674	8.64%	1.83E-634
d.flat	15357	2.41%	2.02E-264	15518	3.48%	4.20E-389	15714	4.79%	2.45E-508
e	19998	3.41%	1.63E-479	20325	5.10%	1.07E-705	20638	6.72%	1.55E-924
f	42662	2.26%	1.21E-655	43090	3.28%	1.33E-969	43556	4.40%	1.01E-1274

TABLE II

EXPERIMENTAL RESULTS: WATERMARKING LUT-BASED TECHNOLOGY MAPPING SOLUTIONS FOR A SET OF ONE SMALL AND SIX INDUSTRIAL DESIGNS. FIRST FOUR COLUMNS CORRESPOND TO THE COLUMNS IN TABLE I. NEXT, THERE ARE FIVE SUBTABLES WITH STRUCTURE IDENTICAL TO THE SUBTABLES IN TABLE I.

the likelihood of coincidence that some other algorithm will retrieve a solution to both the original and watermarked (additionally constrained) specification. The results presented in both tables were collected by averaging the results obtained by augmenting ten different watermarks into each design.

Although the designs evaluated on the MCNC benchmark suite are much smaller than current industrial circuits, we achieved a likelihood of watermarked solution coincidence that is the average smaller than $p < 10^{-13}$ for each circuit given with an average hardware overhead of 4%. In only one out of twenty cases the overhead was larger than 10%. However, in two cases, design watermarking resulted in a negative overhead. Similarly, we obtained $p < 10^{-26}$ with an average overhead of 7.6%.

Obviously, the potential for real applicability of the intellectual property protection technique can be concluded from Table II. This table contains an evaluation of the watermarking strategy for a set of six industrial designs and one small example that shows an interesting anomaly. The watermarking of large examples resulted in a maximal solution coincidence likelihood of $p < 10^{-134}$ with an average hardware overhead of 1.95%, or $p < 10^{-67}$ with an average overhead of 1.19%. Knowing the current computing power possibly available in the industrial environment, such protection is way beyond the most stringent requirements. Note that in all cases the run-time of the optimization program was within $\pm 5\%$ of the program execution run-time for the original statement.

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 [3] to the standard and watermarked set of MCNC benchmark designs. After specifying that 1% and 2% of internal output nodes become pseudo-primary outputs, the MIS suite retrieved average solutions with 2% fewer and 6% more literals, respectively.

VI. CONCLUSION

We have developed the first secure and reliable approach for intellectual property protection of tools and designs in the combinational logic synthesis domain. The approach relies on a novel methodology for design watermarking. Before the synthesis tool is applied, a set of constraints corresponding to a certain copyright message are added to the original design specification. We have presented a set of protocols for effective design watermarking at the multi-level logic minimization and technology mapping level. As one of the key novelties, we identified the importance of standardizing the interpretation of the input for disambiguous constraint augmentation and detection. 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. We demonstrated that the embedded watermarks are relatively hard to delete and hard to find in an arbitrary solution. We applied our approach to the problem of technology mapping for LUT-based FPGAs using a set of benchmark and industrial designs. With a hardware overhead as low as 1.08%, we have achieved a likelihood of design coincidence smaller than 10^{-332} for a standard industrial design of more than 85K gates and more than 6K primary outputs.

REFERENCES

[1] L. Babel, et al. Directed path graph isomorphism. Intl. Workshop on Graph-Theoretic Concepts in Computer Science, pp.395–400, 1995.

[2] V. Bertacco and M. Damiani. The disjunctive decomposition of logic functions. ICCAD, pp.78–82, 1997.

[3] R.K. Brayton, et al. MIS: a multiple-level logic optimization system. IEEE Trans. on CAD, Vol.6, (no.6), pp.1062–81, 1987.

[4] A.E. Caldwell, et al. Effective Iterative Techniques for Fingerprinting Design IP. IEEE Trans. on CAD, Vol.23, (no.2), pp.208–15, 2004.

[5] R. Chapman and T. Durrani. IP protection of DSP algorithms for system on chip implementation. IEEE Trans. on Signal Processing, Vol.48, (no.3), pp.854–61, 2000.

[6] E. Charbon. Hierarchical watermarking in IC design. CICC, pp.295–305, 1998.

[7] E. Charbon and I. Torunoglu. Watermarking layout topologies. ASP-DAC, Vol.1, pp.213–6, 1999.

[8] D. Chen and J. Cong. DAOmap: A Depth-Optimal Area Optimization Mapping Algorithm for FPGA Designs. ICCAD, pp.752–757, 2004.

[9] J. Cong and C. Wu. An improved algorithm for performance optimal technology mapping with retiming in LUT-based FPGA design. ICCAD, pp.572–8, 1996.

[10] J. Cong and Y. Ding. Combinational Logic Synthesis for LUT Based Field Programmable Gate Arrays. IEEE Trans. on Design Automation of Electronic Systems, Vol.1, (no.2), pp.145–204, 1996.

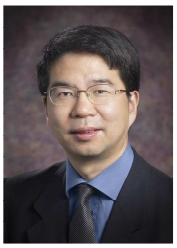
[11] J. Cong and Y.-Y. Hwang. Boolean Matching for LUT-Based Logic Blocks With Applications to Architecture Evaluation and Technology Mapping. IEEE Trans. on CAD, Vol.20, (no.9), pp.1077–90, 2001.

[12] J. Cong and Y. Ding. On Nominal Delay Minimization in LUT-Based FPGA Technology Mapping. FPGA, pp.82–88, 1995.
[13] G. Chen and J. Cong. Simultaneous Logic Decomposition with Technology Mapping in FPGA Designs. FPGA, pp.48–55, 2001.
[14] G. De Micheli. Synthesis and optimization of digital circuits. McGraw-Hill, New York, 1994.
[15] M.R. Garey and D.S. Johnson. Computers and intractability: a guide to the theory of NP-completeness. Freeman, New York, 1979.
[16] G.D. Hachtel and F. Somenzi. Logic synthesis and verification algorithms. Kluwer Academic Publishers, Boston, 1996.
[17] I. Hong and M. Potkonjak. Techniques for intellectual property protection of DSP designs. IEEE ICASSP, 1998.
[18] J.-D. Huang, et al. An iterative area/performance trade-off algorithm for LUT-based FPGA technology mapping. ICCAD, pp.13–17, 1996.
[19] D. Irby, et al. Low level watermarking of VLSI designs for intellectual property protection. Intl. Conf. ASIC/SOC, pp.136–40, 2000.
[20] A.B. Kahng, et al. Robust Intellectual Property Watermarking Methodologies for Physical Design. DAC, 1998.
[21] A.B. Kahng, et al. Constraint-based watermarking techniques for design IP protection. IEEE Trans. on CAD, Vol.20, (no.10), pp.1236–52, 2001.
[22] D. Kirovski and M. Potkonjak. Local Watermarks: Methodology and Application to Behavioral Synthesis. IEEE Trans. on CAD, Vol.22, (no.9), pp.1277–84, 2003.
[23] D. Kirovski, et al. Intellectual Property Protection by Watermarking Combinational Logic Synthesis Solutions. ICCAD, pp.194–198, 1998.
[24] F. Koushanfar, et al. Behavioral Synthesis Techniques for Intellectual Property Protection. ACM TODAES, Vol.10, (no.3), pp.523–45, 2005.
[25] J. Lach, et al. Fingerprinting digital circuits on programmable hardware. Information Hiding Workshop, 1998.
[26] J. Lach, et al. Fingerprinting Techniques for Field Programmable Gate Array Intellectual Property Protection. IEEE Trans. on CAD, Vol.20, (no.10), pp.1253–61, 2001.
[27] S. Liao and S. Devadas. Solving covering problems using LPR-based lower bounds. DAC, pp.117–20, 1997.
[28] S. Meguerdichian and M. Potkonjak. Watermarking while preserving the critical path. DAC, pp.108–11, 2000.
[29] A.J. Menezes, et al. Handbook of applied cryptography. Boca Raton, CRC Press, 1997.
[30] A. Mishchenko, et al. Improvements to Technology Mapping for LUT-Based FPGAs. Intl. Symp. on FPGAs, 2006.
[31] R. Newbould, et al. Watermarking ICs for IP protection. Electronics Letters, Vol.38, (no.6), pp.272–4, 2002.
[32] A. Oliveira. Techniques for the creation of digital watermarks in sequential circuit designs. IEEE Trans. on CAD, Vol.20, (no.9), pp.1101–17, 2001.
[33] A. Rashid, et al. Hierarchical watermarking for protection of DSP filter cores. CICC, pp.39–42, 1999.
[34] G. Wolfe, et al. Watermarking Graph Partitioning Solutions. IEEE Trans. on CAD, Vol.21, (no.10), 2002.
[35] J.L. Wong, et al. Optimization-Intensive Watermarking Techniques for Decision Problems. IEEE Trans. on CAD, Vol.23, (no.1), pp.119–27, 2004.
[36] J.L. Wong, et al. Computational forensic techniques for intellectual property protection. IEEE Trans. on CAD, Vol.23, (no.6), pp.987–94, 2004.
[37] J.L. Wong, et al. Fair watermarking using combinatorial isolation lemmas. IEEE Trans. on CAD, Vol.23, (no.11), pp.1566–74, 2004.

Darko Kirovski received his Ph.D. degree in computer science from the University of California, Los Angeles, in 2001. Since April 2000 he has been a researcher at Microsoft Research. His interests include embedded system design and system security. He received the 1999 Microsoft Graduate Research Fellowship, the 2000 ACM/IEEE Design Automation Conference Graduate Scholarship, the 2001 ACM Outstanding Ph.D. Dissertation Award in Electronic Design Automation, and the Best Paper Award at the ACM Multimedia 2002.

Yean-Yow Hwang received his Ph.D. degree in computer science from the University of California, Los Angeles, in 1999. He has been a member of the R&D staff at Synopsys, Altera, and startup companies since February 2000. His research interests are focused on EDA algorithms on logic minimization, technology mapping, and post-layout re-synthesis and delay minimization.

Miodrag Potkonjak received his Ph.D. degree in electrical engineering and computer science from the University of California, Berkeley in 1991. He is a Professor at the Computer Science Department at UCLA. He received the NSF Career Award, OKAWA Foundation Award, UCLA TRW SEAS Excellence in Teaching Award, and a number of best paper awards. His research interests have focused on CAD and embedded systems, statistical system modeling, computational sensing, and system security.



Jason Cong received his B.S. degree in computer science from Peking University in 1985, his M.S. and Ph.D. degrees in computer science from the University of Illinois at Urbana-Champaign in 1987 and 1990, respectively. He is a professor and the chairman of the Computer Science Department at UCLA and a co-director of the VLSI CAD Laboratory. His research interests include computer-aided design of VLSI circuits and systems, design and synthesis of system-on-a-chip, programmable systems, novel computer architectures, and nano-systems, and highly scalable algorithms. He has published over 230 research papers and led over 30 research projects supported by DARPA, NSF, GSRC, SRC, and a number of industrial sponsors in these areas. He served on technical program committees and executive committees of many conferences. Among other awards, Dr. Cong is a recipient of the NSF Young Investigator Award in 1993, the Northrop Outstanding Junior Faculty Research Award from UCLA in 1993, the ACM/SIGDA Meritorious Service Award in 1998, the 1995 IEEE Trans. on CAD Best Paper Award, the 2005 International Symposium on Physical Design Best Paper Award, and the 2005 ACM Transaction on Design Automation of Electronic Systems Best Paper Award. Dr. Cong served on the ACM SIGDA Advisory Board, the Board of Governors of the IEEE Circuits and Systems Society, and the Technical Advisory Board of a number of EDA and silicon IP companies. He was the founder and president of Aplus Design Technologies, Inc., until it was acquired by Magma Design Automation in 2003. Currently, he serves as the chief technologist advisor at Magma.