

INTELLECTUAL PROPERTY PROTECTION BY WATERMARKING COMBINATIONAL LOGIC SYNTHESIS SOLUTIONS

ABSTRACT

Recently, design reuse has emerged as a dominant integrated system design and integration paradigm. However, 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 has 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, recently, a number of companies have consolidated their efforts towards developing off-the-shelf programmable or application-specific cores (e.g. ARM, LSI Logic). Moreover, it has been estimated that more than half of all ASICs in year 2000 will contain at least one core [Tuc97]. To 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 [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 (as shown in Figure 1). 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 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 in synergy with an arbitrary 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.

The added constraints result in a synthesis trade-off. The more additional constraints, the stronger the proof of authorship, and the higher probability of quality overhead for the logic synthesis solution.

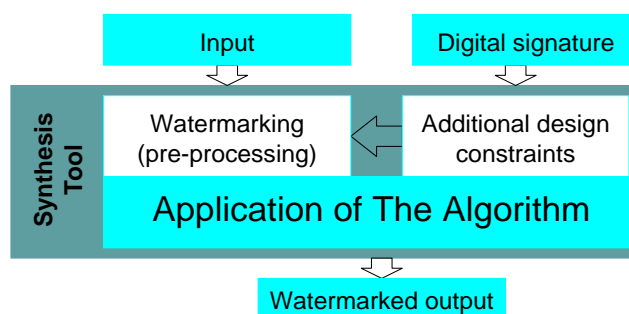


Figure 1: Watermarking logic synthesis solutions transparently to the applied algorithm.

The developed protocols for watermarking solutions of combinational logic synthesis are exactly along the requirements identified in the Strawman [VSI97] proposal of the Development Working Group on Intellectual Property Protection. The recognized desiderata reflects: functionality and timing preservation, transparency to already complex design and verification process, low overhead, provision of a strong and undeniable proof of authorship, flexibility of protection levels with respect to a variety of overhead costs, and persistence. The removal of the embedded watermark should result in a task of the difficulty equal to complete repetition of the specified optimization. In addition to the stated VSI intellectual protection requirements, our approach also provides proportional protection of all parts of the design. We used a set of industrial-strength designs to evaluate the effect of watermarking on the degradation of solution quality. The conducted experiments demonstrated that watermarking in many cases did not induce any performance nor area overhead.

The problem of effective protection of intellectual property in the competitive EDA environment has been addressed at the level of physical design [Kah98] and behavioral specification [Hon98]. In this paper, we present the first techniques for watermarking at the logic synthesis level. Embedding signatures into a design at this 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 watermarking because in that case only the solution to the physical design is protected.

1.1. Motivational Example

We introduce the intellectual property protection approach for logic synthesis solutions and the intuition behind the data hiding protocols by embedding a 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 optimum solution to the technology mapping problem is NP-hard [deM94]. In this simple example, the optimum cover uses six cells. In addition, the cardinality $|S|$ of the set S of all possible 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 ALG_1 returns exactly the same solution as another challenging algorithm ALG_2 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. The added constraints force that the final solution can be retrieved only within a subset S_{sub} of the set S of all solutions to the original logic network specification. 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. The user-specific data, supposed to be augmented into the solution, is a set N of non-redundant numbers. The cardinality of this set is equal to the number of gates in the network. We impose constraints to the problem specification by assigning nodes (the outputs of gates) that have identifiers equivalent to the numbers in the set, to be primary outputs (pseudo-primary outputs in the specification used as input to the synthesis tool). This action poses that the restricted gates have outputs visible in the final library binding. This causality is independent upon the applied library binding algorithm.

For example, assume that the encoded message is $\{3, 8\}$. 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 $S_{3,8}$ of only four solutions of minimal cell cardinality. Therefore, an algorithm applied to this restricted network has to retrieve one of these four solutions. The probability that another algorithm applied to the initial problem specification returns a solution from $S_{3,8}$ is equal to $p = \frac{|S_{3,8}|}{|S|} = \frac{4}{49}$. This probability demonstrates the strength of the proof of authorship. Note, that this example has rather small solution space. As shown in the experimentations, real-life examples have much larger potential for embedding information with little or no loss of solution quality. In these cases, the achieved strengths of authorship are 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 to remove the IP protection? According to the applied protection protocol, how easy is to find someone else's signature in a particular solution? In this paper we answer these questions for the developed watermarking protocols for multi-level logic minimization and technology mapping.

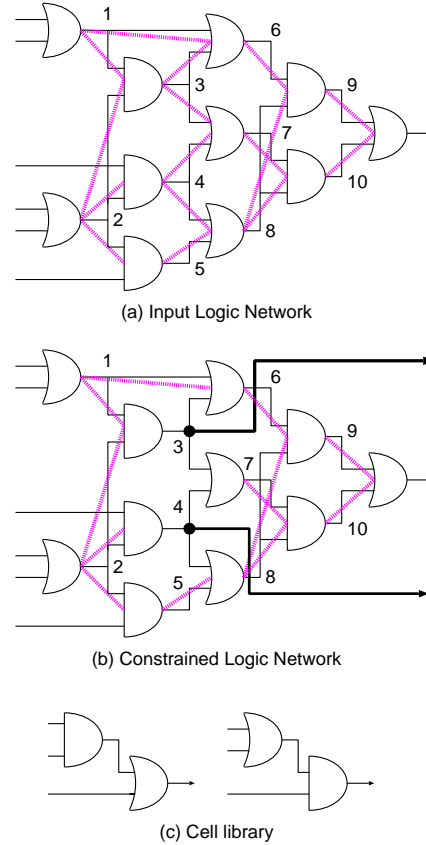


Figure 2: An example of watermarking technology mapping solutions.

2. RELATED WORK

In this section we summarize the literature in two related areas: watermarking techniques and algorithms for combinational logic synthesis tasks: multi-level logic minimization and technology mapping.

Recently, watermarking of artifacts has received a great deal of attention in the research community. The applications facilitating watermarking use the embedded data to track the usage of a particular artifact. Such tracking effectively and inexpensively enables pay-per-use applications and I-commerce of digital goods [Ben96a, Ber96]. There is a wide consensus that intellectual property protection is the prime application of watermarking. A variety of techniques have been proposed for hiding data in still images [Ber96, Cox96]. Several techniques which exploit frequency and time imperfection of the human auditory system have been proposed for watermarking audio content [Ben96a, Cox96]. The AT&T research team developed a number of techniques for watermarking text documents [Ber96]. The video-on-demand research resulted in a suite of approaches for watermarking video [Har97]. As explicitly stated, none of these techniques can be applied to watermark active intellectual property: designs or programs [Ber96]. 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 digital designs has been introduced [Lac98]. It applies encrypted marks to the design in order to support identification of the

design origin and the original recipient.

Combinational logic synthesis has been thoroughly studied. Detailed description of optimization problems and a good survey of minimization techniques and existing non-commercial synthesis frameworks is presented in [Mic94, Hac96]. The targets of latest improvements in combinational logic synthesis are refined covering algorithms [Cou92, Lia97, Gol97], optimizations on networks described using black boxes [Liu97], power optimization [Nar97, Tiw96], etc. Similarly, the research activity in technology mapping has been streamlined towards library- [Gav97, Ped96] and LUT-targeted [Hua96, Con96a] algorithms.

Our approach is the first to define a set of protocols for information hiding into a design at the combinational logic synthesis level. Such watermarking technique provides security against sophisticated reverse engineering attacks [And98], enables concurrently physical layout optimization of a purchased off-the-shelf core and its protection, and exhibits greater potential for adding larger amount of information than the behavioral synthesis watermarking techniques.

3. PRELIMINARIES

In this section we briefly outline the definitions of problems solved in multi-level logic minimization and technology mapping. Then we build the foundation and set of desiderata for our IPP approach, with a brief discussion of the enabled security properties.

3.1. 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 in the logic synthesis are: area, critical path [Bra87], power consumption [Tiw96], testability [Pom95], 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, we adopted that the input to the multi-level minimization process is represented using a logic network. The exact definition of a non-hierarchical combinational logic network is given in [Mic94, Definition 8.2.1].

Given a multi-level logic network constructed using gates exclusively from a given library, the goal of a multi-level logic minimization strategy is to find an equivalent quasi-alternate multi-level representation of the input for which the set of targeted design properties is optimal. A good survey of optimization strategies for multi-level logic minimization is given in [Mic94, Hac96].

3.2. Technology mapping (cell-library binding)

In this step of logic synthesis, the output of the multi-level logic minimization (a logic network) is mapped to a predefined cell library or a network of lookup tables (LUTs). The mapping is performed in a way that the original network is partitioned into subnetworks. The subnetworks are functionally equivalent to the set of cells exclusively selected from a given cell library. The tool developer and the user are considered with 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.

There exist two large classes of algorithms that target this set of problems. The first one is Boolean class where the library cells and the portion of network of interest are represented using Boolean functions. A good survey of approaches from this class of

algorithms is given in [Ben97b]. The second class 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. A good survey of techniques of this class is presented in [Con96a, Mic94, Hac96].

3.3. The Watermarking Desiderata

Recently proposed, Strawman initiative [VSI97] of the Development Working group on Intellectual Property Protection 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 embedded watermark should result in a task of the difficulty equal to the complete re-design 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. DESIGN WATERMARKING AT COMBINATIONAL LOGIC SYNTHESIS LEVELS

We developed an IPP approach which enables the designer or tool developer to embed a signature into the optimized design during execution of several combinational logic synthesis tasks. The key idea is to augment information into the initial specification of the design in such a way that after one of combinational synthesis steps is applied, we have both functionally correct design as well a proof that design is done by the designer and/or the tool.

The synthesis flow which employs watermarking of combinational logic synthesis solutions encompasses several phases illustrated 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 designer's or tool developer's signature. For this purpose, we use a keyed RSA one-way function to generate pseudo-random bits [Men97] which guide the process of iterative gate selection. In the next phase, the outputs of first K gates in the pseudo-random permuted ordering 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 author'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 design specification, retrieves solution which also satisfies the constrained input. In the remainder of this section we explain in detail the key steps in the watermarking protocols and discuss the efficiency and security of the IPP approach.

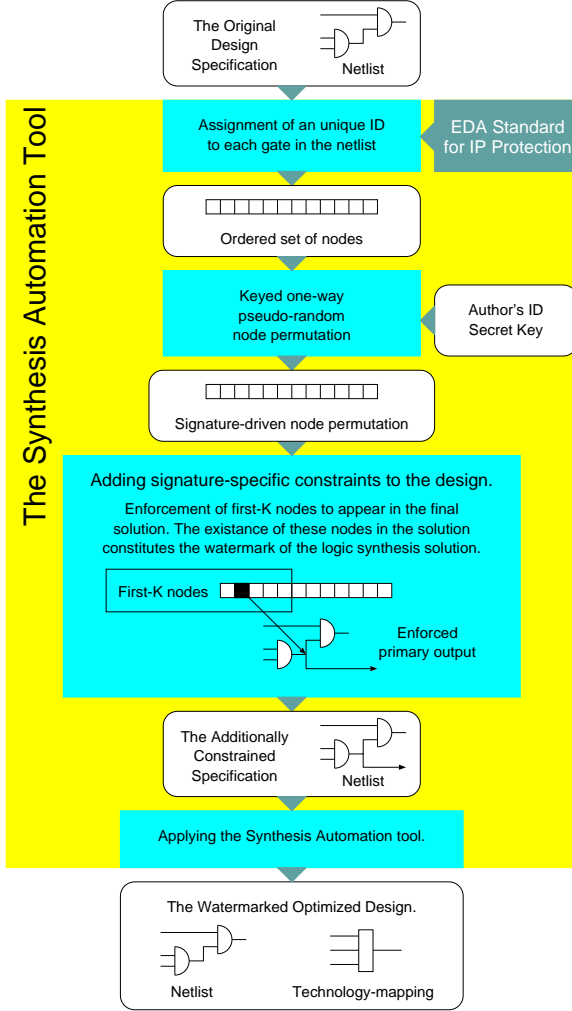


Figure 3: The protocol for hiding information in solutions for multi-level logic optimization and technology mapping.

4.1. Completely Defined 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. The first one is to map the network into a linear array so that cryptographic tools can be directly applied. The second one is to develop fully and uniquely defined watermarking procedure in such a way that the degrees of freedom for potential attackers are maximally reduced.

To disable misinterpretation of this ordering, we propose that an industry assignment standard has to be established. The network has to be numbered in such a way that any two nodes with

different functionality and functionally and timing equivalent transitive fan-ins and fan-outs are assigned different IDs. However, finding whether two nodes are functionally and topologically identical is a hard problem. Its special case, when all gates perform equivalent functions, is equivalent to the directed path graph isomorphism problem. This problem has been listed as open in terms of establishing its complexity [Gar79]. There is no known polynomial time algorithm for this problem. Therefore, we propose a heuristic mapping function, that exploits the node functional and timing properties, to order the nodes in the network. This function is explained using the pseudo-code in Figure 4.

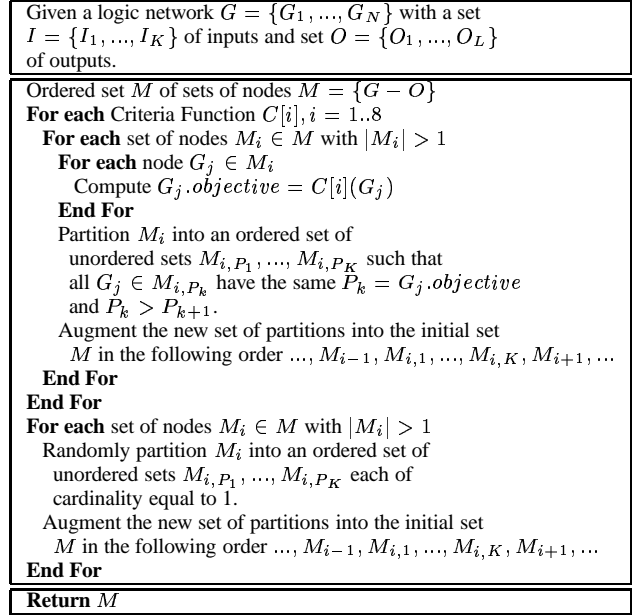


Figure 4: Proposed function for completely defined node ordering.

The mapping function 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 the logic network into an ordered set of node subsets such that each subset contains exactly one node. We propose the following list of eight criteria for node identification:

- C1. 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 .
- C2. The level $LOUT_i$ of node G_i with respect to the output. A node G_i has a level K if the longest reverse path in the logic network from any output to G_i is of cardinality K .
- C3. Number of nodes in the transitive fan-in of G_i at level $K < LIN_i$. The cardinality of this set of criteria at level LIN_i is $LIN_i - 1$.
- C4. Number of nodes in the transitive fan-out of G_i at level $K < LOUT_i$. The cardinality of this set of criteria at level $LOUT_i$ is $LOUT_i - 1$.
- C5. Functionality, fan-in, and fan-out of nodes in the transitive fan-in of G_i at level $K < LIN_i$.
- C6. Functionality, fan-in, and fan-out of nodes in the transitive fan-out of G_i at level $K < LOUT_i$.

- C7. 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 $K < LIN_i$.
- C8. 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 $K < LOU_i$.

An example how nodes are identified using the proposed set of sorting rules is given in Figure 5. 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.

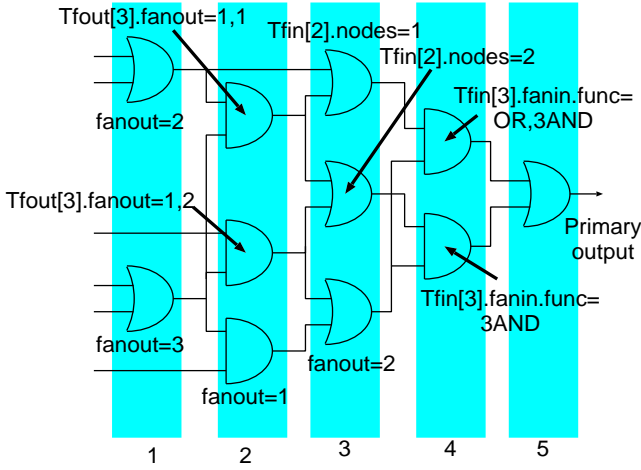


Figure 5: An example of ordering nodes according to the proposed set of sorting criteria.

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 primary outputs (pseudo-primary output). By performing this step, the watermarking routine enforces nodes from the set S to appear as visible in the final technology mapping solution or the variables corresponding to the selected nodes to be computed in the final optimized 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 sequence is keyed using two secret prime numbers. The keys used to drive the randomization process represent the user signature. The result of this phase in the protocol is a pseudo-random signature-specific permutation of network nodes. The first K nodes in the resulting permutation are selected and enforced to appear in the final solution as shown in Figure 3.

4.2.1. Node Selection in Technology Mapping

In the case of technology mapping, 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 nodes which are more likely to appear in the final solution than the remaining nodes. This class of nodes satisfies the following properties. A cone C_G rooted at node G is a subnetwork of internal nodes consisting of G and its predecessors such that if node $F \in C_G$, every path from F to G resides entirely in C_G . The cone C_G is a *fanout-free cone* (FFC) if $fanout(C_G) = fanout(G)$. The *maximal fanout-free cone* (MFFC) at G is the largest FFC rooted at G (see Figure 6). For each node G in the network, there is a unique MFFC rooted at G . Every network can be partitioned into a set of disjoint MFFCs uniquely. It has been shown experimentally that MFFC nodes are more likely to be visible in the final solution [Con96a].

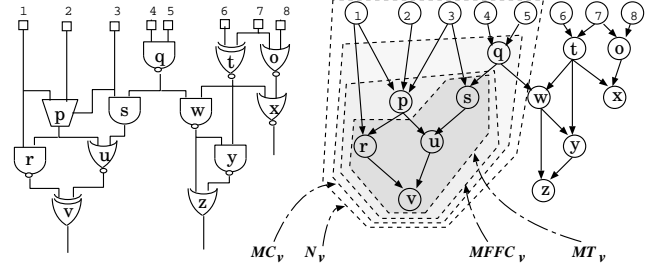


Figure 6: A multi-level logic gate network, its DAG network, and the fanin network, the maximum tree, and the maximum fanout-free cone of v .

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 ($rout$). These ratios have been used as a backbone for our statistical evaluation method. Namely, we compute the likelihood of solution coincidence using the following formula: $p = \left(\frac{rout \cdot F}{rin \cdot T}\right)^{rout \cdot W} \cdot \left(\frac{(1-rout) \cdot F}{(1-rin) \cdot T}\right)^{(1-rout) \cdot W}$, where F is the number of non-primary (including pseudo-primary outputs) 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. Note that, in average, in our experimentations (Section 5), the proof of authorship was actually strengthened by using this approach of evaluating the distribution of probabilities for gate visibility in the final solution of LUT-based technology mapping.

4.2.2. Constraint Encoding in Multi-Level Logic Minimization

In the case of multi-level logic minimization, signature specific constraints are embedded into the logic network in two phases. In the first, already described phase, 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 7.

The network generation is done by iterating the following procedure. During the procedure, the sequence of pseudo-random bits from the previous phase is used to provide a source of undeniable determination. 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 standards may enable customizing the influence of the proof of authorship on the induced overheads. Namely, the application of this procedure may result in cumulative run-time and solution quality overhead.

```

ILN is an input logic network.
PPN is an ordered set of pseudo primary nodes PPN.
PRS = RSABitGenerator(key1, key2), where key1, key2
are the designer 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 ∈ AddedGates
  If G.fanout = null
    ILN.fanout.Add(G)
  End If
End For

```

Figure 7: Proposed function for watermarking multi-level logic minimization solutions using network augmentation.

The additionally constrained original input netlist is now fetched to the optimization algorithm (multi-level logic minimization or technology mapping), which produces the final solution to the problem. 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. To worsen the attacker's position, due to the one-way pseudo-random permutations the attacker cannot conclude how the watermark changes on particular input modification.

4.3. Persistence to Possible Attacks

In order to evaluate the security of the scheme, in the remainder of the section, we discuss the most effective attack scenarios on the proposed protocol.

4.3.1. Watermark Deletion

In the first scenario, 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 leads the attacker to a need for development of 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} . The volume of the Universe is $10^{84} cm^3$. 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 solution.

4.3.2. The attacker finds his/her watermark in an already watermarked solution

There are two scenarios how the attacker can try to find his or her signature in an already watermarked solution (Figure 8). 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.

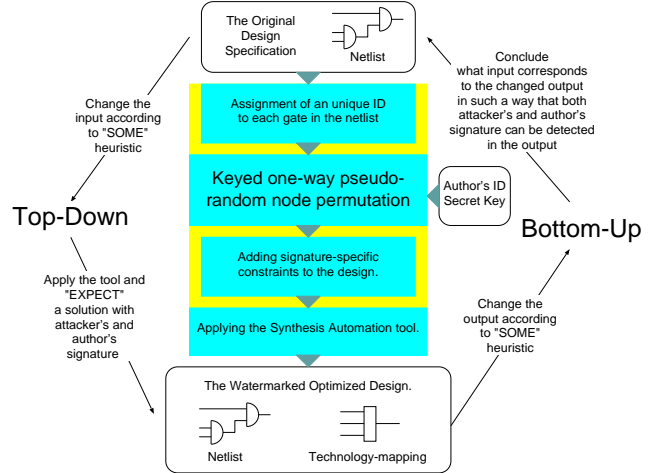


Figure 8: A top-down and bottom-up approach to finding attacker's signature in an already watermarked solution.

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].

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 and six industrial strength designs. For LUT-based technology mapping we used the CutMap algorithm [Con96b].

Tables 1 and 2 show the results obtained when the algorithm was applied to an original and corresponding 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, respectively. The columns in the remaining three-column subtables quantify the performance overhead and quality of watermarking when some percentage (specified in the first row) of non-primary nodes in the original design specification is marked as pseudo-primary. Within each three-column subtable, 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 finally, the third column describes the likelihood of coincidence that some other algorithm will retrieve a solution to both the original and watermarked (additionally constrained) specification.

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 in average equal to $p < 10^{-13}$ (cells with dark background and light font) with average hardware overhead of 4%. In only one out of twenty marked cases the overhead was larger than 10%. However, in two cases design watermarking resulted in negative overhead. Similarly, we obtained $p < 10^{-26}$ (gray cells with dark font) with average overhead of 7.6%.

The potential of real applicability of the IPP technique can be concluded from Table 2. This table contains evaluation of the watermarking strategy for a set of six industrial designs. The watermarking of large examples resulted in average solution coincidence likelihood of $< 10^{-147}$ with average incurred hardware overhead of 1.17% (cells with dark background and light font). Knowing the current computing power possibly available in the industry environment, such protection is way beyond the most strict 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 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 [Bra97] to the standard and watermarked set of MCNC benchmarks. After specifying 1% of non-primary output nodes to become primary, the MIS suite retrieved in average solutions with 2% fewer literals. Similarly, after specifying that 2% of non-primary output nodes are pseudo-primary, the MIS suite retrieved in average solutions with 6% more literals. Due to space limitations we omit presenting detailed description of the experimental results for multi-level logic minimization.

6. CONCLUSION

We have developed the first watermarking-based approach for intellectual property protection 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 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 have identified the importance of standardizing the interpretation of the input for disambiguous signature (constraint) augmentation and detection. We demonstrated that the embedded watermarks are: hard to delete and hard to find in an arbitrary solution. We have applied our approach to the problem of technology mapping for LUT-based FPGAs using a set of benchmark designs. With as low average hardware overhead as 1.17%, we have achieved average likelihood of design coincidence smaller than 10^{-147} for a standard industrial benchmark suite of average size of 26,000 gates.

7. REFERENCES

- [And98] R. Anderson and M. Kuhn. Low cost attacks on tamper resistant devices. Security Protocols, 5th International Workshop Proceedings, pp.125-36, 1998.
- [Ben96a] W. Bender et al. Techniques for data hiding. IBM Systems Journal, Vol.35, no.3-4, pp.313-336, 1996.
- [Ben97b] L. Benini and G. De Micheli. A survey of Boolean matching techniques for library binding. Transactions on Design Automation of Electronic Systems, Vol.2, no.3, pp.193-226, 1997.
- [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. Transactions on Computer-Aided Design of Integrated Circuits and Systems, 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 Yean-Yow Hwang. Structural gate decomposition for depth-optimal technology mapping in LUT-based FPGA design. 33rd Design Automation Conference, pp.726-9, 1996.
- [Cou92] O. Coudert and J.C. Madre. Implicit and incremental computation of primes and essential primes of Boolean functions. 29th Design Automation Conference, pp.36-9, 1992.
- [Cox96] I.J. Cox et al. Secure spread spectrum watermarking for images, audio and video. International 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.
- [Gav97] S. Gavrilov et al. Library-less synthesis for static CMOS combinational logic circuits. International Conference on Computer-Aided Design, pp.658-62, 1997.
- [Gol97] E.I. Goldberg et al. Negative thinking by incremental problem solving: application tounate covering. International Conference on Computer-Aided Design, pp.91-9, 1997.
- [Hac96] G.D. Hachtel and F. Somenzi. Logic synthesis and verification algorithms. Kluwer Academic Publishers, 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. Intellectual Property Protection Techniques for Behavioral Specifications. Unpublished manuscript, 1998.
- [Hua96] J.-D. Huang, J.-Y. Jou, and W.-Z. Shen. An iterative area/performance trade-off algorithm for LUT-based FPGA technology mapping. International Conference on Computer-Aided Design, pp.13-17, 1996.
- [Kah98] A.B. Kahng et al. Robust IP Watermarking Methodologies for Physical Design. To appear, 35th Design Automation Conference, 1998.
- [Lac98] J. Lach, W.H. Mangione-Smith, and M. Potkonjak. Fingerprinting Digital Circuits on Programmable Hardware. To appear, Workshop in Information Hiding, 1998.
- [Lia97] S. Liao and S. Devadas. Solving Covering Problems using LPR-based Lower Bounds. 34th Design Automation Conference, pp.117-120, 1997.
- [Liu97] T.H. Liu et al. Optimizing Designs Containing Black Boxes. 34th 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, CRC Press, 1997.
- [Nar97] U. Narayanan and C.L. Liu. Low-Power Logic Synthesis for XOR based Circuits. International Conference on Computer Aided Design, pp.570-574, 1997.
- [Ped96] M. Pedram, N. Bhat, and E.S. Kuh. Combining technology mapping with layout. VLSI Design, Vol.5, no.2, pp.111-24, 1997.
- [Pom95] I. Pomeranz, and S.M. Reddy. On synthesis-for-testability of combinational logic circuits. 32nd Design Automation Conference, pp.126-32, 1995.
- [Tiw96] V. Tiwari, P. Ashar, and S. Malik. Technology mapping for low power in logic synthesis. Integration, The VLSI Journal, Vol.20, (no.3), 1996.
- [Tuc97] B. Tuck. Core-based methodology maturing in time for mainstream? Computer Design, March 1997.
- [VSI97] VSI Alliance. Fall Worldwide Member Meeting: A Year Of Achievement. Santa Clara, CA, October 1997.

Circuit	PO	Non-PO	Orig	0.50%			1%			2%			4%		
i7	67	439	139	139	0.00%	0.01891	141	1.44%	0.00040	111	-20.14%	1.69E-09	143	2.88%	4.22E-14
i2	1	530	121	121	0.00%	0.01952	123	1.65%	0.00042	127	4.96%	2.44E-07	134	10.74%	1.87E-13
i9	63	471	140	140	0.00%	0.01405	142	1.43%	0.00022	145	3.57%	7.05E-08	152	8.57%	2.33E-14
alu4	8	603	220	220	0.00%	0.04278	221	0.45%	0.00188	226	2.73%	4.69E-06	235	6.82%	5.84E-11
frg2	139	507	302	302	0.00%	0.05632	304	0.66%	0.00337	305	0.99%	1.21E-05	311	2.98%	3.01E-10
rot	107	593	287	287	0.00%	0.02916	287	0.00%	0.00085	291	1.39%	9.38E-07	300	4.53%	2.73E-12
apex6	99	628	242	242	0.00%	0.00960	244	0.83%	0.00010	249	2.89%	1.55E-08	255	5.37%	6.41E-16
C2670	140	716	330	330	0.00%	0.00866	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%	0.00835	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%	0.05434	449	0.67%	0.00301	453	1.57%	1.07E-05	459	2.91%	1.84E-10
i8	81	827	517	515	-0.39%	0.06952	505	-2.32%	0.00399	493	-4.64%	9.88E-06	494	-4.45%	1.06E-10
dal	16	1065	382	385	0.79%	0.00354	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%	0.01424	540	-0.55%	0.00018	545	0.37%	4.11E-08	546	0.55%	1.84E-15
C3540	22	1336	563	563	0.00%	0.00238	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	Non-PO	Orig	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
dal	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 1. Watermarking LUT-based technology mapping solutions for the MCNC benchmark 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. Next, there are seven subtables for different percentage of 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 some other algorithm retrieves a solution, which also contains the embedded watermark.

Circuit	PO	Non-PO	Orig	0.50%			1%			2%		
in1	618	1045	947	947	0.00%	0.0023849	951	0.42%	6.45E-06	959	1.27%	6.84E-11
in2	8010	20238	12091	12177	0.71%	3.54E-70	12245	1.27%	3.31E-138	12424	2.75%	2.08E-268
in3	3934	26574	9825	9942	1.19%	1.59E-86	10017	1.95%	6.86E-171	10237	4.19%	1.48E-330
in4	7452	25727	14996	15080	0.56%	1.21E-68	15175	1.19%	3.55E-135	15357	2.41%	2.02E-264
in5	5142	47470	19338	19492	0.80%	4.79E-124	19700	1.87%	2.13E-244	19998	3.41%	1.63E-479
in6	6832	86058	41721	41991	0.65%	5.28E-168	42172	1.08%	3.87E-333	42662	2.26%	1.21E-655

Circuit	PO	Non-PO	Orig	3%			4%		
in1	618	1045	947	966	2.01%	1.07E-15	968	2.22%	1.39E-20
in2	8010	20238	12091	12593	4.15%	2.98E-393	12768	5.60%	2.2E-509
in3	3934	26574	9825	10462	6.48%	1.22E-486	10674	8.64%	1.83E-634
in4	7452	25727	14996	15518	3.48%	4.20E-389	15714	4.79%	2.45E-508
in5	5142	47470	19338	20325	5.10%	1.07E-705	20638	6.72%	1.55E-924
in6	6832	86058	41721	43090	3.28%	1.33E-969	43556	4.40%	1.01E-1274

Table 2. Watermarking LUT-based technology mapping solutions for a set of one small and five industrial designs. First four columns correspond to the columns in Table 1. Next, there are five subtables with structure identical to the subtables in Table 1.

We guarantee only the order of magnitude for numbers smaller than 2.07E-268.

in1=FOJ1.flat in2=CAMI.flat in3=G4.flat in4=JPEGG.flat in5=COMPAQ_VGA in6=PISCES