Behavioral Synthesis Techniques for Intellectual Property Protection

Inki Hong† ‡ and Miodrag Potkonjak‡
†Synopsys, Inc. Mountain View, CA 94043
‡Computer Science Department, University of California, Los Angeles, CA 90095

Abstract[1]

The economic viability of the reusable core-based design paradigm depends on the development of techniques for intellectual property protection. We introduce the first dynamic watermarking technique for protecting the value of intellectual property of CAD and compilation tools and reusable core components. The essence of the new approach is the addition of a set of design and timing constraints which encodes the author's signature. The constraints are selected in such a way that they result in minimal hardware overhead while embedding the signature which is unique and difficult to detect, remove and forge. We establish the first set of relevant metrics which forms the basis for the quantitative analysis, evaluation, and comparison of watermarking techniques. We develop a generic approach for signature data hiding in designs, which is applicable in conjunction with an arbitrary behavioral synthesis task, such as scheduling, assignment, allocation, and transformations. Error correcting codes are used to augment the protection of the signature data from tampering attempts. On a large set of design examples, studies indicate the effectiveness of the new approach in a sense that the signature data, which are highly resilient, difficult to detect and remove, and yet easy to verify, can be embedded in designs with very low hardware overhead.

1 Introduction

1.1 Motivation

Reusable cores recently emerged as one of the most visible and important components on which future design approaches will be based. Exponential growth of both applications and implementation technology have outpaced the design productivity of the traditional synthesis process. There is a wide consensus that through reuse of optimized cores the demands of applications and ultra large scale integration will be matched. Both industry and academia have expressed a great interest on ways how to conduct core-based designs. For example, all CAD vendors and majority of design houses and silicon foundries joined the Virtual Socket Initiative [7].

There is another wide consensus that viability of the new technology depends on the development of sound mechanisms for intellectual property protection (IPP) [6, 7]. Although there are several forms of protection available for intellectual property in the electronic design automation industry which include patents, copyrights, mask works, trademarks and trade secrets [4], all these methods are insufficient and/or inapplicable for intellectual property protection of reusable cores. In the last few years numerous watermarking or signature hiding techniques have been proposed for image, video, voice and text data [1]. However, these techniques do not provide an adequate solution for design intellectual property protection. Most of these techniques alter a large number of components in the object [1]. While in multimedia this alteration is invisible to human eyes, in designs and CAD tools it will have unacceptable impact on the functionality and correctness.

We propose a new approach which is specifically developed for intellectual property protection of designs and CAD tools. The approach embeds distributed encoded messages in deep structural property of designs at all levels of design process with minimal hardware overhead while maintaining the correctness and functionality of designs. CAD tools are protected in the same way by embedding signatures in designs that they produce. It is based on the key property of design space exploration in behavioral synthesis such that there are numerous competitive solutions. The idea is to select one of the competitive solutions which encodes a signature. Searching for such solution with the embedded signature is difficult, if not an impossible task. However, generating such a design that satisfies the original specified constraints as well as additional watermarking constraints is both easy and time efficient.

1.2 Motivational Example

To illustrate the key ideas behind the new approach, we consider the design shown in Figure 1. The design is a 4th order continued fraction infinite impulse response (CF IIR) filter [3]. Figure 1(a) shows the control data flow graph (CDFG) for the filter. Figure 1(b) presents its scheduled CDFG in 10 control steps.

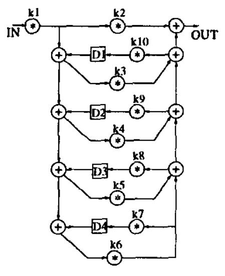
Values that are generated in one control step and used in a later step must be stored in a register during the intermediate control steps. A variable is *live* between the time it is generated (written) and the last use (read) of it. This interval is called the *lifetime* of the variable. Two variables whose lifetimes do not overlap can be stored in the same register. The interval graph [12] is constructed such that for each variable, a node is made in the graph and two nodes are connected if the lifetimes of the corresponding variables overlap. Register assignment can be performed by coloring the interval graph, which is an NP-complete task [5].

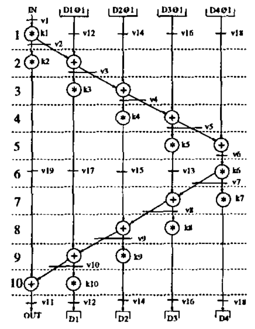
The interval graph for our example is shown in Figure 1(c). Since there are several cliques of size 5 (for example $v1$, $v12$, $v14$, $v16$, $v18$), we need at least 5 colors for the graph. An optimal coloring solution with 5 colors (White, Black, Green, Red, and Yellow) is shown in Figure 1(c). Figure 1(d) shows the resulting datapath and control with 1 adder and 1 multiplier after register assignment are performed. In addition to the optimality in terms of the number of registers, the control and datapath is embedded with a watermark, "A7", which is extremely difficult to detect without knowing the encoding rules. We showed this watermarked design to many designers in industry and researchers in academia and no one was able to detect the embedded watermark.

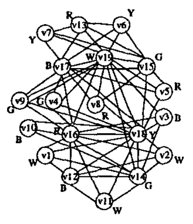
The essence of the new approach is the addition of a set of extra constraints which encodes the author's signature. For the register assignment problem, the extra constraints are imposed such that a set of variables are forced to be stored in different registers which result in extra edges in the interval graph. The interval graph along with the extra edges is shown in Figure 2(a).

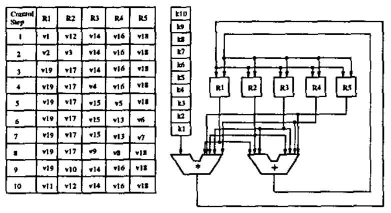
We assume the ASCII encoding for the characters to be used in

[1]This research was supported in part by NSF under grant CCB-9734166.


(a) CDFG


(b) Scheduled CDFG


(c) Colored interval graph



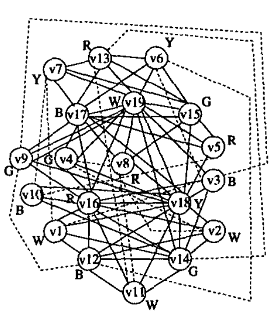
Control Step	R1	R2	R3	R4	R5
1	v1	v12	v14	v16	v18
2	v2	v3	v14	v16	v18
3	v19	v17	v14	v16	v18
4	v19	v17	v4	v16	v18
5	v19	v17	v15	v5	v18
6	v19	v17	v15	v13	v6
7	v19	v17	v15	v13	v7
8	v19	v17	v9	v8	v18
9	v19	v10	v14	v16	v18
10	v11	v12	v14	v16	v18

(d) Control and datapath embedded with a signature "A7"

Figure 1: Watermarking 4th order CF IIR filter

watermarking. 7 bits are used to encode a character in the ASCII code. All the nodes in the interval graph are sorted and numbered in the increasing order of the length of their lifetimes. In the increasing order of node numbers, each node is considered for embedding a single bit. Each consecutive 7 added edges represent a character in the ASCII code. After all the nodes are considered, we repeat the process from the first node in the order. For each bit in the watermark, the terminal node is chosen such that the terminal node number represents the bit value. To embed 1, the terminal node of the added edge should have odd number. To embed 0, the terminal node of the added edge should have even number. The first feasible node in the increasing order of node numbers, starting from the position right after the terminal node of the last edge added, is selected as the terminal node to embed the bit value. This scheme is to add extra edges more evenly in the graph. The added edges with their encoded values for our example are shown in Figure 2(b). They encode "10000010110111" in binary which is equal to "A7" in the ASCII code.

The probability that any register assignment method will result in the same solution with the watermark is extremely low for graphs with reasonably many nodes. For example, the interval graph in Figure 1(c) has 35 unique coloring solutions with 5 colors. Therefore, there is 2.9% chance that any good register assignment method finds the same solution, if we assume uniform probability for all possible 5-color solutions. Considering that the interval graph consists of only 19 nodes, the number is low. Even for this small example, we were able to embed a fairly large signature. In Section 5, we show that there are exponentially many competitive solutions for the graphs with a relatively low number of nodes. We emphasize here that the proposed approach targets and is progressively more effective for large designs.


(a) Colored interval graph with the extra edges added

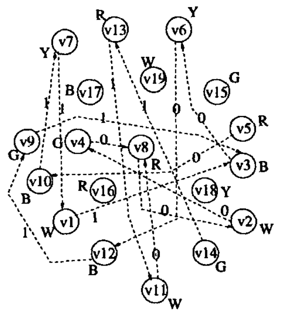

(b) Extra edges with encoded values

Figure 2: Illustrated explanation how the signature "A7" is embedded in the register assignment solution shown in Figure 1(c)

1.3 Research Objectives

We introduce the dynamic watermarking technique for protecting intellectual property of CAD tools and reusable core components, especially targetting behavioral synthesis. The essence of the new approach is the addition of a set of design and timing constraints as a preprocessing design step which encode the signature of the author. The technique is dynamic in the sense that the watermarking constraints are chosen as a preprocessing step before the design is performed and thus influence the final design. The constraints are selected in such a way that they result in the minimal overhead in area, throughput, testability, and/or power consumption, while embedding the signature which is unique and difficult to detect and remove. The idea is based on the key property of design space exploration in behavioral synthesis which consists of several interdependent NP-complete subtasks, where there are many competitive solutions with similar quality.

The proposed technique has a spectrum of important unique properties. The approach provides the quantitative treatment of the intellectual property protection. The technique is transparent to manual and automatic design processes and therefore can be used in conjunction with any available or future set of design tools. The generic technique is applicable to all design steps. It also provides the designers the ability to embed their signatures easily in many different ways. Another novelty is the connection of watermarking to error correcting codes.

We believe that simplicity, strength, low hardware overhead, distributive nature, resilience against tampering and complete transparency will make the proposed technique the method of choice for industrial strength protection of designs, CAD tools, and algorithmic intellectual property. The method also builds conceptual and algorithmic foundation for future research on design intellectual property protection.

1.4 Paper Organization

The rest of the paper is organized in the following way. In the next section we outline the related work. In Section 3, the first set of relevant metrics for the quantitative analysis of watermarking techniques is established. In Section 4, a generic approach for data hiding in design is described. We demonstrate the effectiveness of the approach by applying it to register assignment in Section 5. In Section 6, the approach is also applied to several other behavioral synthesis tasks, such as scheduling and transformations. Finally, we conclude by summarizing the key contributions.

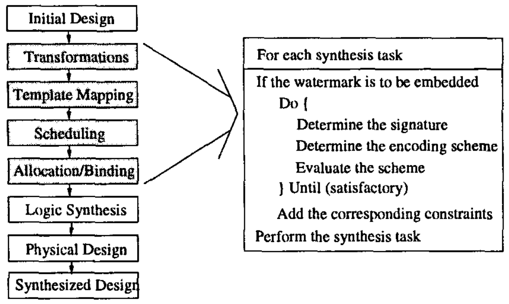


Figure 3: The global flow of the generic approach

2 Related Work

In this Section, we review the most relevant related work on watermarking for CAD and multimedia.

Data watermarking (also known as data hiding) embeds data into digital media for the purpose of identification, annotation, and copyright. Recently, the proliferation of digitized media and the internet revolution are creating a pressing need for copyright enforcement schemes to protect copyright ownership. Several techniques for data hiding in digital images, audios, videos and texts have been developed [1]. Craver et al. discussed a possible attack to any watermarking schemes for multimedia images and proposed a method which is resilient to such attack [2]. The key difference of our proposed approach with those techniques is that all of them embed the watermark in the final object while our approach is incorporated in the design process. In the CAD domain, there have been several proposals for IPP using watermarking at the level of physical layout [9] and logic synthesis [10].

3 Objectives and Metrics for Watermarking

In this Section we lay out the technical foundations required to develop a consistent quantitative approach for intellectual property protection using watermarking techniques. We first analyze the key watermarking properties which imply the quality of the associated data hiding technique. We conclude the Section by summarizing the selected numerical metrics which quantify the effectiveness of a proposed steganography technique for design properties.

3.1 Objectives

While the watermarking techniques are to a certain extent related to subjective criteria about their qualities, we focus our attention on the most relevant objective criteria. In order to be effective, the watermark should exhibit the following properties. Note that these objectives can be mutually conflicting.

- **Correctness of Functionality.** The correctness of functionality should not be affected by the watermark.
- **Low Hardware Overhead.** The watermark should result in low design performance overhead.
- **Transparency.** The addition of the watermark to designs by CAD tools should be completely transparent so that it can be used for any existing CAD tools and designs. The design process is already so involved and conceptually and computationally difficult that the integration of watermarking constraints would result in numerous algorithmic and software difficulties. This implies that watermarking should be done either as preprocessing or postprocessing step.
- **Proof of Authorship.** The watermark should be readily detectable by the owner and law enforcement authorities and unambiguously identify the owner and be accompanied with the convincing proof on its quality.
- **Difficult to Detect.** The watermark should be unobtrusive (perceptually invisible). If thieves could find it, they would solve the most difficult piece of the watermark removal problem. Note that being difficult to detect implies that special knowledge is required to detect the watermark.
- **Resilience.** The watermark must be difficult to remove without significantly degrading the quality of the original design by any techniques which do not have complete knowledge of the design.
- **Proportional Part Protection.** The watermark should be distributed all over the design in order to facilitate the protection of not only the complete design, but also its parts.

3.2 Metrics

From the objectives for the watermark, the following metrics which allow the comparison and evaluation of different watermarking techniques can be identified.

- **Strength of the Authorship Proof.** The probability of *coincidence*, P_C, that the same design with the watermark is produced by any other authors must be minimized. The probability is proportional to the probability that any specific design is produced by a synthesis tool or by a manual design.
- **Resilience.** We numerically define the resilience by the following parameters: (i) the probability that k bits of the watermark is removed by random tampering, e.g., changing a register assignment of one variable in register binding (ii) the number of bits in the watermark, and (iii) the percentage on the number of bits which can be removed while preserving the initially encoded authorship message.
- **Design Metrics Degradation.** The degradation of the design metrics by the watermark must be minimized. The design metrics degradation is quantitatively expressed by a percentage alteration of the relevant design metrics due to the embedding of the watermark.

4 Global Flow for IPP using Watermarking

The generic approach for watermarking designs is shown in Figure 3. The essence of the new approach is the addition of a set of design and timing constraints which encodes the author's signature. The constraints are selected in such a way that they result in the minimal hardware overhead impact while embedding the signature difficult to detect and remove. The selection of the constraints depends on the employed encoding scheme. The selection of the encoding scheme to achieve the goal is a key problem. In the process, the proposed metrics for the quantitative analysis of watermarking techniques are used as guidances.

The technique can be also used for fingerprinting. Distributors of intellectual property such as hardware designs, softwares, documents, and images wish to prevent registered users from releasing unauthorized copies. Fingerprinting allows a distributor to detect any unauthorized copy and trace it back to the user by uniquely watermarking each copy of intellectual properties.

5 Watermarking for Register Assignment

As shown in the motivational example in Section 1, the watermark for register assignment solutions can be embedded in designs in the following steps.

First, the signature data to be embedded should be selected. The signature data should be selected such that they uniquely identify the author as the owner of the design.

The selected signature data need to be encoded so that they can be embedded in the design. Different encoding schemes will result in different quality watermarking solutions. However, by using a cryptographic technique, the encoded message by any encoding schemes can be transformed to a pseudo-random bitstream by an

# of Nodes	P_E	# of Colors	# of Edges Added for k More Colors				
			$k=0$	$k=1$	$k=2$	$k=3$	$k=4$
100	0.1	5	85	195	490	690	795
	0.3	10	100	290	385	685	790
	0.5	16	165	250	305	585	785
200	0.1	8	195	795	1185	1590	2175
	0.3	16	180	605	790	1560	2055
	0.5	26	150	755	950	1380	1815
500	0.1	14	575	2570	3855	5340	7500
	0.3	32	560	1630	3250	4955	5875
	0.5	52	1305	2095	3860	4450	6340
1000	0.1	23	4995	6595	9790	12985	14660
	0.3	55	2995	6495	7995	9980	14585
	0.5	89	850	1495	2975	4500	5930
2000	0.1	38	8855	19990	28505	39865	50010
	0.3	95	7650	18560	27635	36780	45310
	0.5	157	6560	11290	18925	24915	30265

Table 1: Experimental results of watermarking the random graph examples in graph coloring: P_E - edge probability.

encipher. This step results in two advantages. First, it strengthens the proof of authorship by allowing only the designer with the secret key to decipher the signature data. Next, it makes the signature data to look like a random bitstream so that the detection of the signature data by unauthorized users using any statistical analysis becomes harder.

The message is enciphered in the following steps. First, the author's plaintext signature data is applied to MD5 cryptographic hash function. The generated hash is encrypted by the public key of the designer using RSA public key cryptosystem. Using the obtained cipher as an input, RC4 stream cipher generates a pseudo-random keystream. This keystream is combined with the plaintext signature data by a bitwise exclusive-or operation to produce the ciphertext signature data. Finally, the ciphertext signature data are embedded as extra constraints in the design.

We assume same encoding scheme as described in the motivational example in Section 1. To improve the reliability of the proposed intellectual property protection techniques against tampering of design, an error correcting and detection code for the embedded signature are employed. The m-bit error correcting BCH codes [11] have been used to encode the signature data.

Among the metrics, the probabilities of coincidence and tampering for the watermark are especially primary indicators of its protection strength. We note that the use of the terminology "probability" does not follow its exact meaning from mathematics in a rigorous sense. The "probability" in this subsection is rather used as an approximation to the actual probability. We provide the formula to compute the probabilities for random graphs in graph coloring. For other synthesis tasks, the probabilities can be defined and computed similarly. For a random graph $G(n,p)$ with n nodes and edge probability p, suppose c colors are used and k edges are added as watermarking constraints. Let P_C denote the probability of generating the same coloring solution with the signature, given the solution uses the same number of colors. $P_C = (1-\frac{1}{c})^k$. The probability that any register assignment method will result in the same solution with the watermark is extremely low for graphs with reasonably many nodes. Let P_T denote the probability of removing one or more signature bits by changing colors of one node. We approximate the P_T by $P_T = 1-(1-\frac{k}{\frac{n(n-1)(1-p)}{2}})^{\frac{n}{c}-1}$. It is based on the following reasoning. On average there are n/c nodes with the same color. The probability that no signature bit is removed can be approximated by the probability that there are no encoded edges between the n/c nodes with the same color.

From these formula, we note that P_C decreases exponentially as the number of bits in the signature increases. Thus, it is better to use a larger signature to prove the authorship. We also note, however, that P_T increases as the number of bits in the signature increases, which means that it is better to use a smaller signature for preventing tampering. These two requirements are mutually contradictory. This problem can be resolved by using error correcting codes (ECC). We can achieve low P_C and P_T by embedding a large signature with m-bit error correcting codes, where m should increase as the number of bits in the signature increases. The probability of tampering after m-bit error correcting codes are used, is approximated by $\sum_{i=m+1}^{\frac{n}{c}} Pi$, where Pi is the probability that i encoded edges are removed by changing colors of one node. Pi is approximated by $\frac{P_T}{2^i}$.

5.1 Experimental Results

To compare and evaluate the effectiveness and efficiency of the heuristic algorithms, standard test cases from the random graph model as well as real-life designs are used. In the random graph model, for a graph $G(n,p)$ with n nodes, edges are generated independently with probability p between each pair of nodes. Although unlikely to have much relevance to practical applications of graph coloring, such a test bed has the advantage in a sense that behavior of any heuristic on one graph of this type is a good predictor for its behavior on any other [8].

Graph	# of bits in signature	# of bit errors to be corrected	# of parity bits for ECC	# of extra colors used
$G(100,0.1)$	255	25	164	6
$G(100,0.3)$	511	13	117	10
$G(100,0.5)$	1023	7	70	12
$G(200,0.1)$	255	30	184	2
$G(200,0.3)$	511	17	153	5
$G(200,0.5)$	1023	16	160	8
$G(500,0.1)$	511	38	288	1
$G(500,0.3)$	1023	25	245	3
$G(500,0.5)$	2047	19	209	4
$G(1000,0.1)$	1023	25	245	0
$G(1000,0.3)$	2047	20	220	0
$G(1000,0.5)$	4095	25	300	7
$G(2000,0.1)$	2047	20	220	0
$G(2000,0.3)$	8191	15	195	1
$G(2000,0.5)$	16383	10	140	2

Table 2: The overhead for achieving $P_C = 10^{-50}$ and $P_T = 10^{-50}$ for the graphs in Table 1

For another research project, we developed a heuristic for the graph coloring problem which outperforms the best heuristics reported in [8]. The quality of the graph coloring algorithm is provided by several key search guidance techniques. A well known technique of subdividing the problem into an iterative search for an optimal maximally independent set is used to partition the problem. This idea is augmented with a set of global probabilistic least-constraining most-constrained heuristics which boost the algorithm's performance. Extensive experimentation has shown that the algorithm outperforms the state-of-the-art techniques reported in [8] in solution quality with an order of magnitude run-time improvement. The algorithm achieves very high probability of coloring a random, $G(1000,0.5)$ graph with 85 colors in only 5 hours on a Sun Sparc5 workstation (1994 model), compared to average 85.5 colors for average 136 hours on an IBM 3081 which is much faster than the Sparc5 workstation.

Table 1 provides the experimental results for various random graphs. The first column presents the number of nodes in the graph and the second column the edge probability. The third column provides the number of colors used for the original random graph. The fourth to eighth columns presents the number of edges that can be added before k more colors are added, where $k = 0,1,2,3,4,$

respectively. When adding edges, 5 edges were added at a time, which explains the fact that all the numbers in Table 1 are multiples of 5. Since the purpose of this experimentation is not to achieve the minimum possible solution but to generate good quality solutions fast in many times for the demonstration of the effectiveness of our approach, we restrict the running time of the heuristic within a reasonable amount of time. For example, the running time for the graph $G(1000, 0.5)$ is limited to 10 minutes on SUN Sparc5 workstation. In Table 2 we show the overhead for achieving $P_C = 10^{-50}$ and $P_T = 10^{-50}$, in terms of the number of parity bits used for error correcting codes and the number of extra colors. Note that even higher level of protection is attainable with more overhead.

Design	N_V	N_R	# of Edges Added for k More Registers				
			$k=0$	$k=1$	$k=2$	$k=3$	$k=4$
1	48	17	25	35	60	100	150
2	108	26	85	155	400	460	710
3	97	29	60	100	190	340	690
4	67	27	50	120	205	300	415
5	30	16	20	30	55	80	105
6	354	108	300	480	775	1100	1690
7	227	45	180	210	300	515	880
8	200	48	200	510	600	830	1045
9	324	49	95	325	395	800	1250
10	464	122	815	1435	2340	2880	4520
11	827	65	2050	3895	4670	5760	7580
12	1257	45	2895	5845	6475	7980	9220
13	752	104	845	1560	2860	3985	5950
14	1704	128	4865	8160	13520	18910	24235
15	2048	184	3675	6520	9975	13635	18170

Table 3: Experimental results of watermarking 15 real-life designs in register assignment: N_V - # of variables, N_R - # of registers.

We also applied our approach on 15 real-life designs. The designs are numbered in the following order: two GE controllers, two Honda controllers, a wavelet filter, an NEC digital-to-analog converter (DAC), two components of modems, a linear controller for automotive motion control, an LMS audio formatter, a CORDIC, a speech compressor, 2D Wang DCT, 2D Arai DCT, and 2D Lee DCT. We have chosen the small and moderately sized designs to demonstrate the effectiveness of the approach for most difficult examples. As the design is smaller, it is harder to embed large signatures. Table 3 provides the experimental results for the designs. The second column presents the number of variables in designs and the third column the number of registers used for the original design. The fourth to eighth columns presents the number of edges that can be added before k more registers are used, where $k = 0, 1, 2, 3, 4$, respectively. In Table 4 we show the overhead for achieving $P_C = 10^{-50}$ and $P_T = 10^{-50}$, in terms of the number of parity bits used for error correcting codes and the number of extra colors. We assume that the edge probability in all designs is 0.3. It is a moderate assumption since most of the designs result in sparse interval graphs.

All the experimental results show that large watermarks can be embedded in designs while incurring minimal hardware overhead.

6 IPP for Other Behavioral Synthesis Tasks

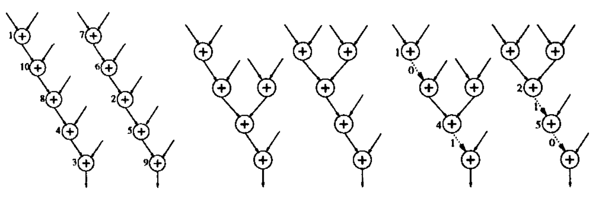
6.1 Transformations

Variables in the original design often disappear after the application of transformations. We embed the author's signature by providing such constraints that particular variables should remain after the application of transformations. For example, we consider associativity transformation. The following encoding scheme can be used: All the operations in a design are randomly numbered. A variable

Design	# of bits in signature	# of bit errors to be corrected	# of parity bits for ECC	# of extra colors used
1	255	20	144	6
2	255	23	156	2
3	255	27	172	3
4	255	30	184	3
5	127	21	98	5
6	511	26	234	2
7	511	28	252	3
8	511	25	225	2
9	1023	24	235	4
10	1023	16	160	1
11	2047	18	198	1
12	2047	20	220	0
13	2047	18	198	2
14	4095	20	240	0
15	8091	18	234	1

Table 4: The overhead for achieving $P_C = 10^{-50}$ and $P_T = 10^{-50}$ for the designs in Table 3.

x from operation x represents 1 if x is even. It represents 0, otherwise. Following the order in the list, for each bit, we search the first variable that can represent the value of the bit.



(a) Original CDFG (b) Watermarked transformed CDFG (c) Encoded variables

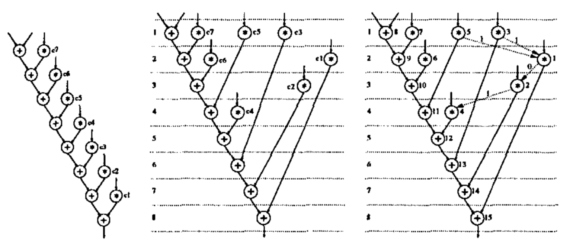
Figure 4: An example of watermarking transformation solutions

Consider a CDFG in Figure 4(a). The available clock cycles are 4 clock cycles. Its transformed CDFG after applying associativity is shown in Figure 4(b). The design is embedded with a signature "6". Figure 5(c) shows how the watermarking is achieved. The number in the left side of each node represents the position in the randomly ordered list. The set of variables to be untouched by transformations is 1,2,4,5, as shown in dotted lines. The set encodes "0110", i.e., "6" in the BCD code.

6.2 Scheduling

The author's signature is embedded by adding a set of constraints such that two operations with no dependencies are forced to be scheduled in one specific order, i.e., an additional edge is added in the *transitive closure* of the CDFG. A transitive closure of a graph $G = (V, E)$ is a graph $G' = (V, E')$, where for every two nodes u and v in V, $(u, v) \in E'$ if $\exists$ a path from the node u to the node v. Whenever an edge is added, the transitive closure of the CDFG must be updated accordingly. In addition, the periods between as soon as possible control step and as late as possible control step that an operation can be scheduled to satisfy the timing constraints must be overlapped in at least 2 control steps for the two operations. Otherwise, adding an edge between the operations is not a new constraint. Whenever an edge is added, the periods for operations must be updated accordingly.

The following encoding scheme can be used: All the operations in a design are sorted and numbered based on the degrees of their



(a) CDFG (b) Watermarked schedule (c) Encoded added edges

Figure 5: An example of watermarking scheduling solutions

scheduling freedom in a decreasing order. The scheduling freedom of an operation is defined to be the length between as soon as possible control step and as late as possible control step that the operation can be scheduled to satisfy the timing constraints. An added edge (x, y) represents 1 if x and y are either both even numbers or both odd numbers. It represents 0, otherwise. Following the order in this sorted list, for each operation x, we search the first operation y from the beginning of the list that an edge (x, y) can be added and represent the desired value.

Consider a computational structure represented by a CDFG in Figure 5(a). The available clock cycles are 6 clock cycles. Its scheduled CDFG is shown in Figure 5(b). The design is embedded with a signature "7". Figure 5(c) shows how the watermarking is achieved. The number in the right side of each node represents the position in the sorted list. The set of additional edges is (1,2), (2,4), (3,1), (5,1), as shown in dotted lines. The set encodes "0111", i.e., "7" in the BCD code.

7 Invalidating False Claims of Ownership

Consider the following scenario. Alice has a design D which is watermarked by the proposed watermarking scheme. Bob purchases the design D from Alice. There are three cases to consider.

- **Finding Unintended Signatures:** Bob tries to find his signature from the design D. Bob claims that D is his design because D has both his and Alice's signatures. In this case, one with more meaningful, longer and stronger signature will be a winner. It is believed to be a very difficult task to find any meaningful signature from the design. It is much easier to embed the signature in a preprocessing step. Therefore, Alice can protect her authorship by embedding strong watermark.
- **Embedding Unauthorized Signatures:** Bob embeds his signature to the design D and claims that D is his design. In this case, Alice can easily prove that the design belongs to her. Alice's design has only her signature, but Bob's design has both his and Alice's signatures. Therefore, Alice can prove that it is her design. Note that this type of attack is similar to one described in [2].
- **Tampering Original Signatures:** In this case, Bob tries to tamper Alice's signature by applying local change. There are at least three reasons that discourage this kind of attack. First, these local changes can result in worse design with more hardware overhead. Secondly, error correcting and detection codes allow the design to withstand significant amount of attack. Thirdly, once Bob modifies the design, he has to perform all synthesis tasks below the affected part. For example, if the register-transfer level design is modified, then the logic synthesis and physical design should be performed which can make the final design completely different from the original.

8 Conclusion

We introduced the first dynamic watermarking technique for protecting intellectual property of CAD tools and reusable core components. The essence of the new approach is the addition of a set of design and timing constraints which encodes the author's signature. The constraints are selected in such a way that they result in the minimal hardware overhead impact while embedding the signature difficult to detect and remove.

We established the first set of relevant metrics for the quantitative analysis of watermarking techniques, which forms the basis for the quantitative analysis, evaluation, and comparison of watermarking techniques. We developed a generic approach for steganography and use it as a basis for the derivation of a spectrum of techniques for signature data hiding in designs. Techniques have been applied to several behavioral synthesis tasks, such as scheduling, register assignment, and transformations.

On a large set of design examples, studies indicated the effectiveness of the new approach in a sense that highly resilient, difficult to detect and remove, yet easy to verify signatures are embedded in the designs with very low hardware overhead.

REFERENCES

[1] W. Bender, D. Gruhl, N. Morimoto, and A. Lu. Techniques for data hiding. *IBM Systems Journal*, 35(3&4):313–336, 1996.

[2] S. Craver, N. Memon, B. L. Yeo, and M. M. Yeung. Can invisible watermarks resolve rightful ownerships? Technical report, IBM Research Technical Report RC 20509, 1996.

[3] R.E. Crochiere and A.V. Oppenheim. Analysis of linear digital networks. *Proceedings of the IEEE*, 63(4):581–595, 1975.

[4] D. Fernandez. Intellectual property protection in the EDA industry. In *Design Automation Conference*, pages 161–163, 1994.

[5] M.R. Garey and D.S. Johnson. *Computer and Intractability: A Guide to the theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, 1979.

[6] E. Girczyc and S. Carlson. Increasing design quality and engineering productivity through design reuse. In *Design Automation Conference*, pages 48–53, 1993.

[7] Virtual Socket Initiative. http://www.vsi.org.

[8] D.S. Johnson, C.R. Aragon, L.A. McGeoch, and C. Schevon. Optimization by simulated annealing: an experimental evaluation; II. graph coloring and number partitioning. *Operations Research*, 39(3):378–406, 1991.

[9] A. B. Kahng, et al. Robust IP Watermarking Methodologies for Physical Design. In *Design Automation Conference*, pages 782–787, 1998.

[10] D. Kirovski, Y.-Y. Hwang, M. Potkonjak, and J. Cong. Intellectual Property Protection by Watermarking Combinational Logic Synthesis Solutions. In *International Conference on Computer-Aided Design*, pages 194–198, 1998.

[11] S. Lin and D.J. Costello. *Error Control Coding*. Prentice Hall, 1983.

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