Watermarking Integer Linear Programming Solutions

Seapahn Megerian
Computer Science Department
University of California Los Angeles

seapahn@cs.ucla.edu

Milenko Drinić
Computer Science Department
University of California Los Angeles

milenko@cs.ucla.edu

Miodrag Potkonjak
Computer Science Department
University of California Los Angeles

miodrag@cs.ucla.edu

ABSTRACT

Linear programming (LP) in its many forms has proven to be an indispensable tool for expressing and solving optimization problems in numerous domains. We propose the first set of generic watermarking techniques for integer-LP (ILP). The proof of authorship by watermarking is achieved by introducing additional constraints to limit the solution space and can be used as effective means of intellectual property protection (IPP) and authentication. We classify and analyze the types of constraints in the ILP watermarking domain and show how ILP formulations provide more degrees of freedom for embedding signatures than other existing approaches. To demonstrate the effectiveness of the proposed ILP watermarking techniques, the generic discussion is further concretized using two examples, namely Satisfiability and Scheduling.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection -- *Digital Watermarking*

General Terms

Algorithms, Economics, Theory, Legal Aspects.

Keywords

Digital Watermarking, Intellectual Property Protection

1. INTRODUCTION

Recently, the advent of modularized hardware and software libraries and core-based system-on-chip (SOC) design methodologies fueled a flurry of research focused on intellectual property (IP) protection (IPP). One of the techniques that has been shown to be extremely effective for IPP is watermarking (WM) due to its potential simplicity, strong authorship proofs, flexibility, and low overhead. Watermarking techniques utilize the fact that generally, tasks used in IP design rely on computationally intractable optimization problems having numerous solutions of similar quality. By introducing additional constraints corresponding to the author's signature, they limit the solution space and hence make a convincing authorship proof if a particular solution can be shown to belong to the constrained solution space. Numerous Watermarking methods have been proposed for different stages of the design process, including system synthesis, logic synthesis, technology mapping, and physical design [Kah98, Hon95, Meg00].

Linear programming is an efficient method for minimizing a linear objective function under linear inequality constraints. Integer linear programming (ILP) is a special form of linear programming with the additional restriction that variables are integers. While any linear programming instance can be solved in polynomial time, solving ILP is NP-hard. Still, numerous effective techniques and packages have been developed that solve fairly large ILP instances in reasonable run times. The popularity of ILP as a backbone to many optimization algorithms has prompted the need for effective ILP watermarking techniques. In this paper we classify and discuss a generic set of techniques that can be used to watermark the solutions of ILP formulations. In addition to the generic-level discussion, we illustrate the methods using two specific problems, namely Satisfiability (SAT) and Scheduling. In addition to their wide range of applications, these two problems are well suited to complement our discussions since SAT is decision oriented while Scheduling is an optimization problem. Although problem-specific watermarking techniques have already been proposed for both SAT and Scheduling, we show the effectiveness and power of ILP watermarking techniques and discuss why they can be superior.

1.1. An ILP Watermarking Example

To motivate the ILP watermarking discussion, we use a very simple SAT example. We watermark the solution of the simple instance by embedding signature constraints in its corresponding ILP formulation in two ways. Consider the five-variable formula:

$$F = (x_1 + x_2 + x_3) \cdot (x_4 + x_5)$$

There are a total of 21 possible satisfying truth assignments out of the 32 total assignments for F. We assume that any of the valid solutions are equally likely to occur so the probability that a specific solution is selected as the final answer is 21/32. The ILP formulation corresponding to the SAT instance represented by F can be stated as:

$$
\begin{aligned}
\text{Minimize:} \quad & \varnothing \\
\text{Where:} \quad & x_1 + x_2 + x_3 && \geq 1 \\
& x_4 + x_5 && \geq 1
\end{aligned}
$$

For the sake of brevity, we ignore the complemented variables and assume that all variables are defined as 0-1 integers. A watermark is essentially a set of characteristics imposed in the final ILP solution by limiting the solution space. The following formulation demonstrates one such constraint using the ILP objective function:

$$
\begin{aligned}
\text{Maximize:} \quad & x_1 - x_2 - x_3 + x_4 - x_5 \\
\text{Where:} \quad & x_1 + x_2 + x_3 && \geq 1 \\
& x_4 + x_5 && \geq 1
\end{aligned}
$$

This new ILP formulation still represents the original SAT instance corresponding to F. The new objective function signature here is obtained using the pseudo-random bit string "10010" (see 5.2). The new solution space has been reduced to 1 out of the original 21 valid assignments, an indication of the WM strength. Note that for this example, this is the strongest possible watermarking solution.

Another method for imposing watermarking constraints in ILP formulations is the use of augmented variables. Consider the addition of a new integer variable x_a in the original formulation:

Maximize:	x_a	
Where:	$x_1 + x_2 + x_3 - x_a$	≥ 1
	$x_4 + x_5 - x_a$	≥ 1
	x_a	≥ 0

It is easy to verify that the new formulation has 3 solutions of equal quality. Such use of augmented variables in order to impose new and complex relationships among unrelated constraints is one of the main advantages of using ILP WM. Previously proposed WM methods do not provide the flexibility and power of ILP techniques to impose complex relationships among groups of constraints and variables. Generally, practical solution instances are much larger and provide a rich environment for embedding long signatures by simultaneously applying several different WM techniques.

1.2.   Paper Organization

The remainder of this paper is organized as follows: Section 2 contains the related work. Section 3 contains the preliminary discussion including definitions, notations, and assumptions. Section 4 presents the generic ILP watermarking techniques and constraint classifications. Sections 5 and 6 present the details of ILP watermarking specific to SAT and Scheduling with supporting experimental results and analysis, followed by the conclusion.

2. RELATED WORK

Imperfections of the human visual and auditory system have been used as the basis for hiding data in still images, audio, and video [Bor96]. However, watermarking IP where no errors are allowed, such as designs and programs, require different techniques. Protocols for watermarking active IP have been developed at the physical level [Kah98], technology mapping and template matching [Meg00], and behavioral synthesis [Hon95]. Qu et al. address the questions of the quality of the solution subspace and the difficulty of finding a solution from such subspace in [Qu00].

Boolean Satisfiability and Scheduling have numerous applications in CAD and other domains. Many design problems such as test pattern generation, circuit delay computation, logic optimization, and combinational equivalence checking can be formulated as SAT problems [Mar00]. An example of a heuristics for solving SAT problem is presented in [Bay96]. Summery of important classical scheduling theory results for real-time computing can be found in [Sta95] and [Pau89]. An example of an ILP-based formulation for Scheduling is presented in [Geb91].

3. PRELIMINARIES
3.1.   SAT

Let $U=\{u_1, u_2,..., u_m\}$ be a set of Boolean variables. A *clause* over U is a set of literals that represents their disjunction. A clause is *satisfied* by a truth assignment if and only if at least one of its members is true under that assignment. The Satisfiability problem is hence defined as [Gar79]:

Problem: *SATISFIABILITY (SAT)*
Instance: *A set U of variables; A collection C of clauses over U.*
Question: *Is there a satisfying truth assignment of C ?*

Usually SAT problems are stated in the from of a formula. Section 1.1 contains an example of a SAT formula and its corresponding ILP formulation. In our notation, x_i' indicates the logical complement of the variable x_i. For simplicity, we use the same names to represent the Boolean SAT variables and the ILP variables. We assume that all SAT ILP variables are defined as 0-1 integers unless otherwise specified.

3.2.   Scheduling

We use the synchronous data flow (SDF) as our primary computational model [Lee87]. SDF is a special case of general data flow, in which the number of data samples produced or consumed by each node is specified a priori. The syntax of a targeted computation is defined as a hierarchical control-data flow graph (CDFG) [Rab91]. The CDFG represents the computation as a flow graph, with nodes, data edges, and control edges. The semantics underlying the syntax of the CDFG format is that of the SDF flow model. Operation scheduling is the process of partitioning the set of operations in a CDFG into groups such that the operations in the same group are executed concurrently in the same control step. The task of operation scheduling is to determine the amount of resources necessary for execution of operations in a CDFG.

The two general approaches for scheduling are heuristic-based methods [Pau89] and ILP [Geb91]. In this paper, we restrict our attention to ILP for time-constrained scheduling, where for a given number of control steps, the set of arithmetic and logical operations in a CDFG is partitioned such that each partition corresponds to a single control step. The goal of scheduling here is to minimize the number of functional units necessary for the execution of all operations in the CDFG. The problem can formally be stated as:

Problem: *Time-Constrained SCHEDULING of a CDFG.*
Instance : *A CDFG C, a time constraint t, parameter n.*
Question: *Does there exist a Scheduling of C into t control steps, such that the number of operations from C in each control step is less than n and all precedence constraints in C are satisfied?*

The problem of Time-Constrained Scheduling of a CDFG is computationally intractable as it is essentially the same problem as precedence constrained scheduling, a well known NP-complete problem [Gar79]. To present the ILP formulation of the time-constrained scheduling problem [Geb91], we define variables x_{ij} as follows:

$$x_{i,j} = \begin{cases} 1 & \text{if operation } j \text{ is scheduled in control step } i. \\ 0 & \text{Otherwise.} \end{cases}$$

$1 \leq i \leq n$, $1 \leq j \leq m$, where n is the number of the given clock cycles and m is the number of operations in the CDFG. The condition that each operation has to be scheduled in exactly one control step is expressed through the equation:

$$(1) \qquad \forall j \quad \sum_{i=1}^{n} x_{i,j} = 1$$

This condition is usually replaced with a tighter condition taking the earliest and the latest possible times when an operation can be scheduled into consideration.

$$(2) \qquad \forall j \quad \sum_{i=k}^{l} x_{i,j} = 1$$

where $k = ASAP(x_{i,j}) \ \forall i$, and $l = ALAP(x_{i,j}) \ \forall i$. $ASAP(x_{i,j})$ and $ALAP(x_{i,j})$ refer to *as soon as possible* and *as late as possible* heuristic algorithms for scheduling. The data flow in the CDFG is restricted with precedence constraints, formulated as:

$$(3) \qquad \sum_{i=1}^{n}(n-(i+1))x_{i,j_1} \geq 1 + \sum_{i=1}^{n}(n-(i+1))x_{i,j_2}$$

where j_1 is the operation in the CDFG whose output is used as an input by the operation j_2. We also define variables x_i as:

$$(4) \qquad \forall i \quad x_i = \sum_{j=1}^{m} x_{i,j}$$

The variable x_i represents the actual number of functional units for each control step i. The problem of minimizing the number of functional units necessary for execution of the CDFG under the above constraints can now be formulated as:

Minimize: y
Where: $x_i < y$ $\quad\quad\quad \forall i$

This definition can trivially be extended to the allocation problem, separating the conditions for number of functional units to different types of units.

4. GENERAL ILP WATERMARKING

The global flow for ILP watermarking is shown in Figure 1. The main strategy is to add a number of additional constraints and try to satisfy as many of them as possible. The watermarking constraint selection is made according to a pseudorandom number generated in correspondence with a signature. The constraint encoding procedure yields sufficiently randomized constraints to hinder the detection and forging of signatures. Figure 2 shows the encoding process: SHA is a one-way hash function, RSA is a public key encryption system, and RC4 is a stream cipher (SHA and RSA are used in the PGP software package). RC4 outputs a cryptographically strong pseudorandom bit-stream that the constraint encoding step uses to generate the WM constraints. Verification of a signature involves both testing for signature presence and matching the signature to the text file of the alleged owner. Demonstrating that the signature is in the solution is done by showing an unusual presence of enough WM constraints. A signature can be shown to correspond to the text file and the owner's public key using PGP.

The types of methods used to embed watermarks in ILP formulations have three main classifications: (i) *augmented variable definition*, (ii) *objective function*, and (iii) *ILP constraints*. In the following subsections, we provide an overview of each of these methods.

4.1. Variable Definitions

The variables used in ILP formulation of optimization and decision problems fall into two categories: (i) *natural* and (ii) *augmented*. *Natural* variables are variables that directly represent aspects of the original problem. *Augmented* variables on the other hand, have no apparent relationship to the problem and often do not have any physical meaning. The ability to define and use augmented variables in ILP formulations is a powerful method of imposing additional constraints in order to improve watermarking strength, which is often not possible with other methods.

Augmented variables enable the imposition of complex relationships among the natural variables and constraints. These new dependencies are key factors that make the ILP based watermarking methods more powerful than previously proposed algorithm-specific techniques. Section 5 and 6 provide several examples to demonstrate how augmented variables can be used to impose specific watermarking relationships among independent constraints.

4.2. Objective Function

Generally, each ILP formulation is composed of an objective function F_0 and a set of constraints. Any number of additional objective functions can be used for signature encoding. The general objective function F_{WM} used in WM ILP has the following structure:

$$F_{WM} = \alpha_0 F_0 + \alpha_1 F_1 + \alpha_2 F_2 + \alpha_3 F_3 + \dots$$

where F_i ($i>0$) denote augmented watermarking objective functions and coefficients α_i are real valued constants.
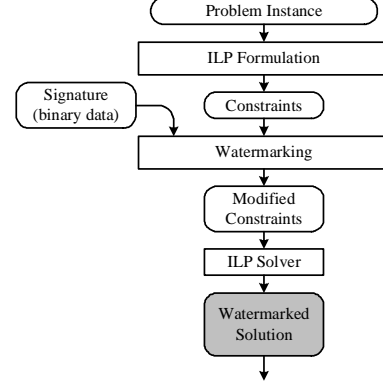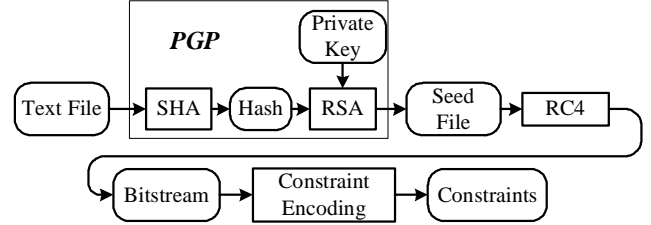

Figure 1. Global Flow For ILP Watermarking

Figure 2. ILP Watermarking - Constraint Encoding

Note that in optimization intensive applications, $\alpha_0 >> \alpha_i$ for $i>0$ since the quality of the final solution must not be allowed to degrade. The augmented objective functions F_i can be any arbitrary linear function of the natural and augmented ILP variables. Problem specific attributes can guide the construction of such functions in order to improve WM strength and preserve solution quality. Certain decision problems such as SAT, do not have any inherent objective functions in their corresponding ILP formulation and hence provide the most flexibility for embedding such signatures.

4.3. ILP Constraints

Modifying the original ILP constraints is the third and most broad method of embedding WM constraints. To ensure solution correctness, almost all such techniques manipulate the original set of ILP constraints such that the new resulting constraints are more restrictive. As the case with ILP variables, ILP constraints can be either *natural* or *augmented*. Natural constraints arise from the problem specification and care must be taken to ensure that they are not violated as a result of the watermarking process. Augmented constraints do not generally have any obvious relation to the problem at hand and can provide great freedom in embedding signatures. Furthermore, each type of constraint can be classified as *positive* or *negative*. A *positive* constraint specifies the value or range of values that each variable should be assigned to, while a *negative* constraint specifies undesirable assignments.

5. ILP WATERMARKING – SAT
5.1. SAT WM: Variable Definitions

Normally, there is little freedom in redefining the natural set of variables in SAT. However, augmented variables allow the imposition of complex relationships among the existing and/or augmented constraints. Consider the following constraints:

$$x_1 + x_2 + x_5 + x_9 + x_{21} \geq 3$$
$$x_3 + x_7 + x_{11} + x_{17} \geq 2$$

- Define augmented integer variable x_a and modify constraints:

$$x_1 + x_2 + x_5 + x_9 + x_{21} - x_a = 3$$
$$x_3 + x_7 + x_{11} + x_{17} - x_a = 2$$
$$x_a \geq 0$$

In this simple example, the new variable x_a defines a new relationship between the two original constraints. So, although previously each constraint could be satisfied independently, here, there is a new underlying dependency that relates seemingly unrelated constraints. This notion can further be used to include multiple sets of constraints and multiple augmented variables. However, care must be taken to ensure that the additional constraints do not cause a satisfiable formula to become unsatisfiable and vice versa.

5.2.   SAT WM: Objective Functions

As stated earlier, SAT is purely a decision problem and hence does not pose any inherent objective functions. This additional degree of freedom provides powerful means of encoding SAT watermarking constraints. For instance, consider a SAT formulation with 10 variables and a 10-bit pseudo-random bit sequence (signature) S:

$$S = \text{"1011001011"}$$

- Define a new SAT ILP objective function F_1 by mapping each variables x_i to a bit $s_i \in S$. We add the variable x_i if the corresponding bit $b_i = 1$ and subtract x_i if $b_i = 0$:

$$F_1 = x_1 - x_2 + x_3 + x_4 - x_5 - x_6 + x_7 - x_8 + x_9 + x_{10}$$

Maximizing F_1 in the ILP in effect is an attempt to preserve the signature S in the form of variable assignments. Here, if all variables are assigned according to the bits in S, then the WM strength is maximized. However, such an assignment may not be a valid solution. The scheme can be enhanced using the first- and second-order heuristic objective functions presented in [Qu99]. The functions essentially calculate the likelihood of particular assignments for each variable in a valid solution. This likelihood information can be used to improve the mapping process of the signature string to the WM objective function, thus increasing the WM strength. Specifically, the heuristics can be used to decide if each bit $b_i \in S$ should be mapped to the variable x_i or its complement x_i' (+ or – in the objective function) in order to increase the likelihood that assigning variables according to S would result in a valid solution.

5.3.   SAT WM: ILP Constraints

Let us now consider ILP constraint manipulation techniques for watermarking. Each *natural* constraints in the ILP corresponds to a clause in the SAT formula. Consider the constraint:

$$x_1 + x_2 + x_5 + x_9 + x_{21} \geq 1$$

- We can reduce the solution space by *removing variables* from such constraints. This has the same effect as *deleting literals* from corresponding clauses that is discussed in [Qu99]. For example, we can replace the constraint above with the new constraint assuming that the new constraint does not make the formula unsatisfiable:

$$x_2 + x_5 + x_9 \geq 1$$

- We can further limit the solution space by adding new *augmented ILP constraints*. In addition to adding new constraints similar to the ones above (see *adding clauses* in [Qu99]) we may opt to impose special characteristics on groups of variables such as:

$$x_3 + x_4' + x_9 + x_{11}' + x_{17} \geq 3$$
$$x_1' + x_5 + x_6 + x_{12} + x_{20} \geq 4$$
$$x_1 + x_2 + x_5 + x_9 \leq 2$$

Augmented constraints can be combined with augmented variables to produce even more signature embedding options.

5.4.   SAT Experimental Results

We compared the performance of the ILP WM approach to the optimization intensive SAT-solver results presented in [Qu99]. We use the same set of DIMACS benchmark instances. Table 1 lists the DIMACS benchmarks and the number of variables and clauses for each instance. The "Opt. Bits" column shows the number of watermarked bits as reported in [Qu99]. For each instance, we embedded a comparable size random signature in each instance and used LP_SOLVE, a freely available (GNU) mixed integer linear program solver. In all cases, the ILP solver produced the watermarked results with the run-times reported in minutes in column six, on a 333 MHz Sun Ultra 5 Workstation with 128 MB RAM. It is expected that larger instances can be watermarked equally effectively using more sophisticated commercial ILP solvers.

Table 1. SAT WM: ILP vs. Optimization Intensive Technique

Instance	Vars	Clauses	Opt. Bits	ILP Bits	ILP Run time
ii8a1.cnf	66	186	1400	1500	1
ii8a2.cnf	180	800	3100	3400	5
ii8a3.cnf	264	1552	6700	6500	3
ii8a4.cnf	396	2798	3900	4600	7
ii8b1.cnf	336	2068	6800	7200	11
ii8b2.cnf	576	4088	8900	8800	35
ii8c1.cnf	510	3065	7500	7900	22

6.   ILP WATERMARKING – SCHEDULING

Formulating scheduling as an ILP problem opens new dimensions of imposing watermarking constraints that are impossible or difficult to achieve using other methods. In this section we present a few examples to demonstrate how generic ILP watermarking can apply to Scheduling.

6.1.   Scheduling WM: Variable Definitions

- Define augmented variables $x_{i,j}$ that satisfy the equation $\sum_{i=k}^{l} x_{i,j} = 1$ and impose a number of precedence constraints expressed using equations of type (3). The new variables are not added to equations for variable x_i as they are not affecting the quality of the final solution. Here, we exploit a degree of freedoms specific to ILP formulations of scheduling that does not have a physical interpretation.

- Define an augmented variable x_i corresponding to the addition of a new control step. The direct consequence is the quality of the final solution. As shown in Table 2, this technique does not reduce the number of solutions and it should be used in conjunction with other techniques. However, the feasible solutions are altered as a consequence of the constraints and which can serve as a proof of authorship. The only exception is when this technique enables the reduction of number of functional units necessary for execution of operations in a CDFG.

- Define an augmented variable z_i,

$$z_i = x_{i_1,j_1} + x_{i_2,j_2} + \ldots + x_{i_g,j_g} + x_{i_1} + x_{i_2} + \ldots + x_{i_h}$$

under the constraint that $a_i \leq z_i \leq b_i$, , where a_i and b_i are specified positive integers. Note that this type of constraint does not have any real interpretations in the Scheduling domain, yet provides another degree of freedom for watermarking solutions. Many more constraints of this nature with varying structures can be imposed.

6.2. Scheduling WM: Objective Functions

- Introduce correlation constraints for variables x_i:

$$x_{i_1} + x_{i_2} + \ldots + x_{i_g} = x_{i_{g+1}} + x_{i_{g+2}} + \ldots + x_{i_{g+h}}$$

This technique directly affects the objective function. The introduced correlation does not have any direct association to the scheduling problem.

- Maximize α, the weighted total number of satisfied constraints, $\sum_{i=1}^{g} w_j c_i = \alpha \qquad j \in C$, where C is the set of constraints.

Various watermarking constraints introduced in this section can be applied alone or in conjunction to one or more of the other techniques. The ILP solver may be unable to solve the problem under all posed constraints. Here, we introduce another objective function which will allow the ILP solver to satisfy as many WM constraints as possible, while still obtaining a good quality solution.

6.3. Scheduling WM: ILP Constraints

- Fix variable $x_{i,j}$ to 1: $x_{i,j} = 1$, and $x_{g,j} = 0$ for $g \neq i$.

For solution feasibility, the range of i is in [*ASAP*, *ALAP*]. This technique corresponds to fixing an operation from CDFG to a particular control step.

- Fix variable $x_{i,j}$ to 0: $x_{i_1,j} = 0, x_{i_2,j} = 0, \ldots, x_{i_h,j} = 0, k \leq i_i \leq l$.

This constraint in effect forbids an operation from being scheduled in one or more control steps and is thus a *negative* constraint.

- Enforce minimum distance between two variables with existing precedence constraint:

$$\sum_{i=1}^{n}(n-(i+1))x_{i,j_1} \geq d_{\min} + \sum_{i=1}^{n}(n-(i+1))x_{i,j_2}$$

Minimum distance $d_{\min}$, where $d_{\min}$ is a positive integer, represents the minimum number of clock cycles that j_1 is scheduled before j_2.

- Enforcing maximum distance between two variables with existing precedence constraint: The constraints in this case have essentially the same form as the previous case with the exception that the positive integer variable $d_{\max}$ is used instead of the d_{min} and the inequality is '$\leq$' instead of '$\geq$'. Variable $d_{\max}$ represents the maximum number of clock cycles that j_1 is scheduled before j_2.

- Enforcing minimum distance between two variables without existing precedence constraint:

$$\sum_{i=1}^{n}(n-(i+1))x_{i,j_1} \geq d_{\min} + \sum_{i=1}^{n}(n-(i+1))x_{i,j_2}$$

Variables that do not have explicit precedence constraints but have implicit precedence constraints as a result of transitivity of preceding relations are constrained by this technique in such a way that the implicit order is preserved.

- Enforcing maximum distance between two variables without existing precedence constraint: Implicit precedence constraint restricts this technique to cases where $d_{\max}$ is greater than the critical path between j_1 and j_2.

6.4. Impacts of WM on Solution Space

Encoding the signature into constraints is the final step in embedding the signature in the ILP formulation. This step relies on the selection of WM techniques and their associated parameters. We use the first block of bits of a signature bit-stream for technique-
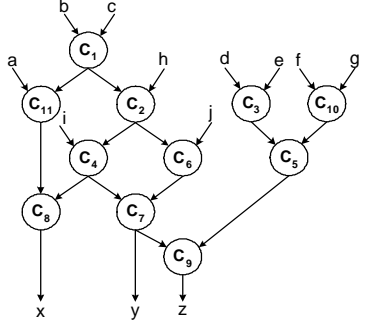
selection and the rest for parameter selection. For each of the presented techniques, we attempt to embed the WM such that after satisfying all constraints, there is still a non-zero probability that a valid solution exists. For example, if a variable $x_{i,j}$ is being fixed to 1, the applied procedure must choose i in the range [*ASAP*, *ALAP*]. Similarly, if the critical path in the CDFG between two variables is more than $d_{\max}$ while enforcing a maximum distance constraint, then the solution to the problem becomes infeasible. Due to the intractability of Scheduling, the proposed procedure cannot guarantee that the solution will exist. Thus, such decisions almost always rely on problem specific heuristics. One attempt at such a heuristic is the α maximization technique presented in Section 6.2, that relaxes the constraints and enables easier solution search for the ILP solver. The weights for each constraint are inversely proportional to the ratio of their resulting average solution space reduction.

Figure 3. A CDFG Example

Consider the CDFG shown in Figure 3, where variable $x_{i,j}$ corresponds to operation C_j and variable x_i corresponds to the control step i. Suppose that the required number of control steps for the execution of this CDFG is 8. Here, the minimum number of functional units required for execution is 3. Table 2 lists the corresponding *ASAP* and *ALAP* assignment for each operation.

Table 2. ASAP/ALAP Scheduling for the CDFG in Figure 3.

OP.	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}
ASAP	1	2	1	3	2	3	4	4	5	1	2
ALAP	4	5	6	6	7	6	7	8	8	6	7

There are 44,715 valid operation-to-control-step (scheduling) assignments of the CDFG in Figure 3. Let us consider the 36-bit sequence 39D6B4C86$_{hex}$ as our signature. If we only use the "fix variable $x_{i,j}$ to 1" technique then our encoding will produce:

$$x_{7,8}=1; \qquad x_{i,8} = 0, i \neq 7$$
$$x_{6,11}=1; \qquad x_{i,11} = 0, i \neq 6$$
$$x_{2,5}=1; \qquad x_{i,5} = 0, i \neq 2$$
$$x_{7,9}=1; \qquad x_{i,9} = 0, i \neq 7$$

We use the first 5 bits of the sequence for selection of the particular variable $x_{i,j}$. Index j is obtained by the formula $j = (code)_{\mod 11} + 1$, where *code* refers to the decimal number corresponding to the first 5 bits of the sequence. The reason for using 5 bits instead of 4, the minimal number, is to make the selection as fair as possible [Qu00]. Usage of only 4 bits would make the likelihood of selecting the first 6 variables twice as large as the rest. Index i is selected similarly, using 4 bits, and normalizing the number to the range of feasible solutions (in the [*ASAP*, *ALAP*] range). If the sequence decoding results in the selection of a variable that is already scheduled, that selection is discarded. After applying the first constraint, the possible number of solutions is reduced to 12,515, after the second constraint the number of solutions drops to 2,793, and after the next two constraints, to 29 and 7 respectively.

If we select the "fix variable $x_{i,j}$ to 0" technique from Section 6.3, using the same encoding as in the previous case ($x_{7,8}=0$; $x_{6,11} = 0$; $x_{2,5} = 0$; and $x_{7,9} = 0$), the number of possible solutions is reduced at

a much slower rate. After applying the first constraint, the number of solutions drops to 32,200, after the second constraint, to 28,437, and after the last two constraints to 28,251 and 24,137 respectively.

Clearly, different methods of converting the signature to WM constraints are possible. We can expect that the "fix variable $x_{i,j}$ to 1" reduces the solution space more than the "fix variable $x_{i,j}$ to 0". Due to the nature of Scheduling ILP formulations, the *positive* constraint of setting a variable to 1 forces several more to 0. The solution space reduction caused by other techniques can be seen in Table 4. The number of possible solutions and minimal number of functional units needed for execution of all operations of the CDFG in Figure 3 is shown in Table 3, as the allowed maximum number of control cycles is changed. For some techniques, the length of the signature in our example is not sufficient to show enough correlation between the technique and the change in solution space size. The impact of the remaining techniques highly depends on the selected variables and the actual enforced constraints.

Table 3. Impact of Increasing Cycles on Number of Solutions

	Total number of cycles available					
	6	7	8	9	10	11
F. Units	2	2	2	2	2	1
Solutions	155	4,175	44,715	257,506	1,435,227	6,944

Table 4. Impact of Watermarking on Solution Space

Technique	WM Sequence [bits]			
	9	**18**	**27**	**36**
Define new variable $x_{i,j}$	-	44715	-	39012
Fix variable $x_{i,j} = 1$	12515	2793	29	7
Fix variable $x_{i,j} = 0$	32200	28437	28251	24237
4	16030	16030	4354	184
5	36438	36438	30685	30305
6	-	44715	-	823
7	-	44715	-	44647

4 – Min distance between variables with existing precedence.
5 – Max distance between variables with existing precedence.
6 – Min distance between variables without existing precedence.
7 – Max distance between variables without existing precedence.

Table 5. Scheduling WM Experimental Results

Application		4K bits signature		16K bits signature	
Name	**# of Ops**	P_c	**Perf. Overhead**	P_c	**Perf. Overhead**
G721	758	10^{-20}	0.2%	10^{-49}	0.7%
Epic	872	10^{-29}	0.1%	10^{-85}	0.9%
PEGWIT	658	10^{-23}	0.8%	10^{-46}	1.6%
PGP	1755	10^{-76}	0.3%	10^{-178}	1.2%
GSM	802	10^{-44}	0.9%	10^{-125}	1.3%
JPEG.c	1422	10^{-68}	0.3%	10^{-289}	2.5%
MPEG2.d	1372	10^{-47}	0.2%	N/A	N/A

6.5.   Scheduling Experimental Results

To quantify the performance of the ILP-based watermarking approach for Scheduling, we compared the performance to the localized WM methodology presented in [Kir99]. Table 5 lists the instances used and the number of operations in each CDFG. The next set of results are for embedding 4K bit- and 16K-bit random signatures. The P_c column shows the calculated likelihood of coincidence as defined in [Kir99]. It estimates the ratio of the number of high quality solutions in the constrained watermarked space with respect to the non-constrained solution space, which can be used as

an approximation for WM strength. Note that the exact calculation of this ratio is intractable due to the exponentially-increasing size of the solution space. The performance overhead column lists the run-time overhead in solving the watermarked ILP formulation vs. the original formulation. In most cases, the ILP-based method was able to produce the specified WM solutions fairly rapidly. However, the limitation of our ILP solver (LP_SOLVE) became apparent on larger instances such as MPEG2.d.

7. CONCLUSION

We presented a set of generic watermarking techniques for ILP solutions. We classified the general types of constraints in the ILP watermarking domain and discussed how each can be used to embed a signature. We also provided specific examples and detailed discussion from the SAT and Scheduling domains and showed several tradeoffs and problem specific attributes in each case. Experimental results indicate that ILP watermarking methods have tremendous potentials for embedding strong watermarks with little overhead.

8. REFERENCES

[Bay96] R.J. Bayardo and R. Schrag. "*Using CSP Look-Back Techniques To Solve Exceptionally Hard SAT Instances.*" Principles and Practice of Constraint Programming, pp.46-60, 1996.

[Bor96] A.G. Bors and I. Pitas. "Image Watermarking Using DCT Domain Constraints." *International Conference on Image Processing*, Vol.3, pp.231-244, 1996.

[Geb91] C.H. Gebotys, and M.I. Elmasry. "*Simultaneous Scheduling And Allocation For Cost Constrained Optimal Architectural Synthesis.*" Proceedings of IEEE Design Automation Conferecnce, pp. 2-7, 1991.

[Gar79] M. R. Garey and D. S. Johnson. *Computers And Intractability: A Guide To The Theory Of NP-Completeness*. W.H. Freeman, 1979.

[Hon95] I. Hong, M. Potkonjak. "*Behavioral Synthesis Techniques for Intellectual Property Protection.*" DAC, pp. 849-854, 1999.

[Kah98] A. Kahng, et al. "*Watermarking Techniques For Intellectual Property Protection.*" DAC, pp. 776-781, 1998.

[Kir99] D. Kirovski, and M. Potkonjak. "*Localized Watermarking: Methodology And Application To Operation Scheduling.*" IEEE International Conference on Computer-Aided Design, pp.596-9, 1999.

[Lee87] E.A. Lee, D.G. Messerschmitt. "*Synchronous Dataflow.*" Procs. of the IEEE, Vol.75, (no.9), pp.1235-45, 1987.

[Mar00] J.P. Marques-Silva and K.A. Sakallah. "*Boolean Satisfiability in Electronic Design Automation.*" DAC, pp. 675–680. 2000.

[Meg00] S. Meguerdichian and M. Potkonjak. "*Watermarking While Preserving The Critical Path.*" DAC, pp. 108-111, June 2000.

[Pau89] P.G. Paulin and J.P. Knight. "*Force-Directed Scheduling For The Behavioral Synthesis Of ASICs.*" IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol.8, pp.661-679, 1989.

[Qu99] G. Qu, J.L. Wong, and M. Potkonjak. "*Optimization-Intensive Watermarking Techniques For Decision Problems.*" 36th ACM/IEEE Design Automation Conference Proceedings, pp. 33-36, June 1999.

[Qu00] G. Qu, J.L. Wong, M. Potkonjak. "*Fair Watermarking Techniques.*" Procs of Design Automation Conference, pp.55-60, 2000

[Sta95] J. Stankovic, et al. "*Implications Of Classical Scheduling Results For Real-Time Systems.*" IEEE Computer, Vol.28, pp. 16-25, 1995.

[Rab91] J. Rabaey, et. al. "*Fast Prototyping Of Data Path Intensive Architectures.*" IEEE Design & Test of Computers, Vol.8, pp.40-51, 1991.