Watermarking Graph Partitioning Solutions

Greg Wolfe, Jennifer L. Wong, and Miodrag Potkonjak
Computer Science Department, University of California, Los Angeles, CA 90095-1596

ABSTRACT

Trends in the semiconductor industry towards extensive design and code reuse motivate a need for adequate Intellectual Property Protection (IPP) schemes. We offer a new general IPP scheme called *constraint-based watermarking* and analyize it in the context of the graph partitioning problem. Graph partitioning is a critical optimization problem that has many applications, particularly in the semiconductor design process. Our IPP technique for graph partitioning *watermarks* solutions to graph partitioning problems so that they carry an author's signature. Our technique is transparent to the actual CAD tool which does the partitioning. Our technique produces solutions that have very low quality degradation levels, yet carry signatures that are convincingly unambiguous, extremely unlikely to be present by coincidence, and difficult to detect or remove without completely resolving the partitioning problem.

1. INTRODUCTION

The exponential growth of VLSI design integration has led to an explosive proliferation of reusable core-based designs. This, in turn, motivates a need for effective and efficient IPP schemes and tools. We present a general *constraint-based watermarking* scheme and analyize it in the context of graph partitioning. Graph partitioning is a critical optimization problem that has many applications, particularly in the semiconductor design process [1].

Our IPP technique for graph partitioning *watermarks* solutions to graph partitioning problems so that they carry an author's signature. The general constraint-based watermarking approach maps an author's signature into a set of constraints and then modifies the partitioning objective function so that a disproportionate number of these constraints are satisfied. This, however, is only a skeleton of the process since the types of constraints that are selected (*constraint types*) and the tactic by which we encourage a disproportionate number of the constraints to be satisfied (*watermarking tactic*) can vary greatly. We developed, implemented, and evaluated five separate schemes that differ from each other solely in these choices.

We introduce our approach using a small example. Consider the graph of Figure 2. We will call this graph G16. It has 16 vertices and 31 edges. It was created randomly by specifying that there are 16 vertices and that Each potential edge will occur with a probability of 0.25. Our goal is to demonstrate that, for even a graph this small, it is possible to watermark solutions of the graph partitioning problem. We also show, in general, that the potential for watermarking exists by demonstrating what happens to the number of solutions of various qualities when certain constraints are enforced. For the sake of this example the graph partitioning problem is formally defined as follows:

Problem: MIN k-WAY BALANCED GRAPH PARTITION Instance: Graph $G = (V,E)$ Solution: A partition of V into k disjoint sets $F = C_1,...,C_k$ with $\lvert C_i\rvert = \frac{\lvert V\rvert}{k}$ for $i = 1,..,k$. Measure: The sum of the weight of edges between the disjoint sets. This measure is called the *edge-cut* of the graph.

Figure 1: Graph partitioning problem.

Variations of this problem allow for weighted vertices or edges, hyperedges rather than edges, and relaxed balance constraints. Finding a solution to any of these problems with minimum edge-cut is NP-hard[8]. For this example, we are concerned with 2-way exactly balanced graph partitioning.

The core idea behind our watermarking technique is to select a set of constraints that correspond to our watermark and then to find a solution to the problem that satisfies a large number of these constraints. We can do this by preprocessing on a problem instance and then running the partitioner (any partitioner) on the modified instance.

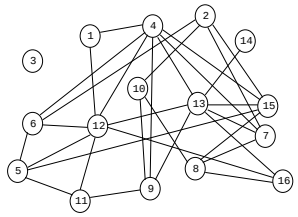

Figure 2: G16: A graph with 16 vertices and 31 edges.

For this illustrative example, our constraints are of the type "vertex v_1 and vertex v_2 must be on the same side of the partition". We will enforce this by merging the selected vertices as a preprocessing step. Therefore, we face a simple tradeoff between the number of constraints added and the quality of our solutions. If, for each constraint, v_1 and v_2 are simply selected randomly from the set of vertices, then the probability of each constraint having occurred in some solution by coincidence alone is $\frac{1}{2}$. Since these probabilities are nearly independent of each other, the probability of X constraints all occurring in a solution by coincidence is approximately $\frac{1}{2}^X$.

We display an exhaustive list of the number of solutions of various qualities, Figure 3. The vertical axis represents edge-cut values and the horizontal axis represents number of solutions on a logarithmic scale. The outermost curve shows the number of solutions that have various edge-cuts. From this curve one can see that the min cut is 9, that the max cut is 25. It is important to observe that for this graph there is only one solution with a cut of the minimum size 9. Thus it is unreasonable to expect to find a watermarked solution which also has cut value of 9.

The other curves correspond to the results of progressively merging pair after pair of vertices together. The pairs of vertices contracted for this example are listed in order in Figure 3. These pairs

DAC 2001, June18-22,2001, Las Vegas, NV, USA.

encode a signature. They were selected using a cryptographically strong pseudorandom number generator seeded in a way that will be discussed later. After the first three of these pairs are contracted, the min cut is 10, the max cut is 23, and there are 37 different solutions with an edge-cut of 13. All of these solutions satisfy all three constraints and hence there is at most a one in eight chance of coming up with one of these solutions by coincidence. Hence a partitioner that returned the partition with an edge-cut of 10 would yield a high quality, watermarked solution.

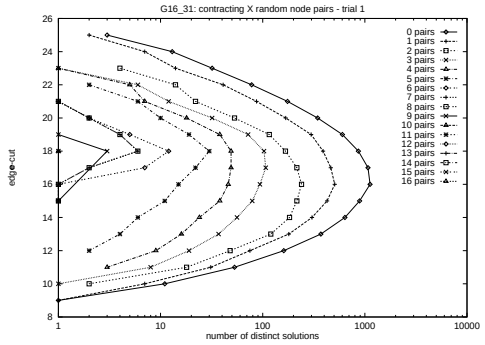

Figure 3: Number of distinct partitioning solutions of the graph G16 with particular edge-cuts as the following pairs of vertices are merged together: (16,14), (6,2), (16,4), (9,8), (5,16), (9,4), (11,10), (9,4), (15,16), (9,7), (2,3), (13,5), (13,14), (10,12), (14,3), (9,8).

2. RELATED WORK

The graph partitioning problem is ubiquitous in many fields of computer science and engineering. It has important applications in areas ranging from work-load balancing in parallel programming, to database storage and in particular to VLSI design and CAD techniques [1].

Throughout the process of VLSI circuit design and synthesis graph partitioning plays a key role. It has applications in system design, behavioral synthesis, system-level synthesis, packaging, rapid prototyping, and testing. The graph partitioning problem is NP-complete. Therefore many heuristic methods are proposed to find high quality partitions.

Watermarking is a form of information hidding that embeds information into an instance of some media. This information is useful for the purpose of identification, annotation, and copyright. The proliferation of digitized media and the prominence of the Internet are creating a mounting need for copyright enforcement schemes to protect ownership. Hence, watermarking of digitized media is becoming increasingly common [3, 15]. In the last three years, a number of watermarking-based IPP techniques have been propsosed including [9, 10, 11, 13, 14].

Several cryptographic techniques are useful to the first step of our watermarking approach, that of finding a set of constraints for the instance of the optimization problem that is to be watermarked. The specific method we use involves the cryptographic hash function MD5 , the public-key cryptosystem RSA, and a stream cipher which may be equivalent to the stream cipher RC4. We use the PGP software package for MD5 and RSA calculations [12].

3. OBJECTIVES AND METRICS

Watermarks should satisfy the following properties: low overhead, strong proof of ownership, hard to find ghost signatures, transparency, difficult to detect, difficult to forge signatures and tamperproof [16].

The objectives of watermarking optimization problems motivate several metrics. These metrics allow the measurement, comparison, and evaluation of watermarks on various problem instances.

Design metric degradation. For minimization problems quality degradation is the watermarked solution's quality over an unmarked solution's quality minus 1.

Strength of authorship proof. The probability P_c that a solution to an optimization problem that was not watermarked by a author coincidentally contains that author's signature must be convincingly low. What should be considered "convincingly unlikely" is very subjective. Probabilities in the range of 10^{-3} to 10^{-12} may arguably be acceptable. If brute force attacks to find ghost signatures are possible, probabilities as low as 2^{-56} may be necessary.

Resiliency metrics. Specific resiliency metrics can be defined for how well a watermark holds up to specific kinds of limited tampering attacks.

4. APPROACH

The approach for watermarking solutions for the graph partitioning problem is shown in Figure 4. The general strategy is to define a number of costraints of some type and then to satisfy a disproportionate number of them. The type of constraints and the tactics by which they are encouraged to be satisfied are discussed later in this section. All of them have in common the selection of a set of vertices or (hyper)edges. The pseudorandom selection process is cryptographically seeded with the owner's signature. This, of course, can be done in a wide variety of ways and can perhaps even be expanded and improved upon to allow additional features like group signatures [5, 6] and undeniable signatures [4].

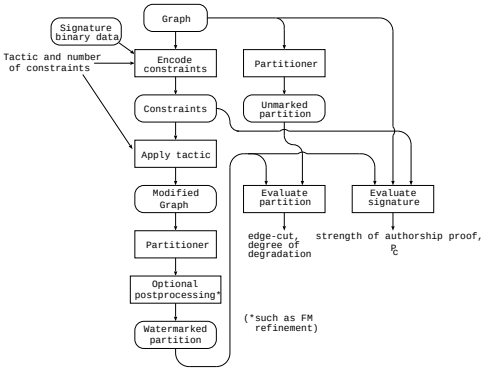

Figure 4: How to watermark: The watermarking process.

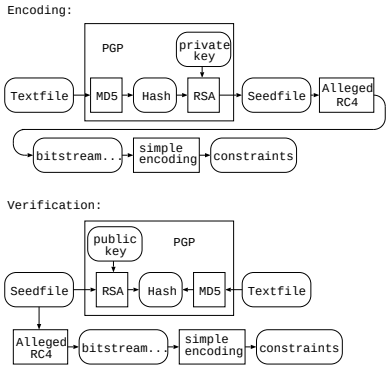

Figure 5: How to watermark: Here we encode constraints.

The current process we use for generating constraints is shown in Figure 4. It is done in this way so that the encoding scheme yields sufficiently randomized constraints and so that it is difficult to detect signatures and to forge signatures. The figure shows both the

encoding process and the process by which one verifies that a signature is present. MD5 is a one-way hash function. RSA is a public key encryption system. "Alleged RC4" is a stream cipher. We use MD5 and RSA only from within the PGP software package. The bit-stream that is the output of "alleged RC4" is a cryptographicly strong pseudorandom bit-stream. The "simple encoding" box uses it for tasks like choosing a pseudorandom vertex and (hyper)edge.

To verify a signature, one must show both that the signature is present in the partitioning solution and that the signature corresponds to the text file and the pgp public key of the supposed owner. Demonstrating that the signature is in the partitioning solution is done by showing that enough of the signature's constraints are satisfied to be unusual. One can show that the signature corresponds to the text file and the owner's public key by running PGP.

The protocol for deciding what RSA keys and text files are used is unspecified. If there is any "degree of freedom" in their selection, then a brute force attack may be able to find ghost signatures. In order for this attack to be computationally difficult, P_c must be sufficiently small. $P_c \leq 2^{-56}$ is likely good enough.

Watermarks are added by defining a set of constraints that correspond to the watermark and then finding a solution that satisfies a sufficiently disproportionate number of these constraints. The success of this endevour can be measured by the amount of design metric degradation and the strength of authorship proof. There are many different types of constraints that can be defined. Additionally, there are many tactics by which these constraints can be imposed in such a way that it is likely that solutions will satisfy a disproportionate number of them. In the introduction we discussed constraining pairs of vertices to be on the same side of the partition by merging them together. Here we discuss that tactic as well as four others. Note that all five tactics work equally well on either graphs or hypergraphs. Later, in the experimental results section, we show how they actually perform. The choice of which tactic to use and how many constraints to add can make the difference between poor and excellent watermarking results.

The constraints imposed by each tactic are completely independent of each other. Because of this, the same constraint may occur several times. In this case it is either satisfied many times or broken many times.

Because all of the constraints are chosen independently, P_c, the probability of a solution carrying an author's watermark purely by coincidence, can be computed by a simple binomial. P_c is a metric for the strength of authorship proof. Let C be the number of constraints imposed, b be the number of these that are *not* satisfied, and p be the probability of a constraint being satisfied purely by coincidence. Let X be a random variable that represents how many of the C constraints were not satisfied. Now, P_c = the probability that b or less of C constraints are not satisfied by coincidence = $P(X \leq b) = \sum_{i=0}^{b}(\binom{C}{i} \cdot (p)^{C-i} \cdot (1-p)^{i})$.

Overestimating the value p is acceptable, since this will always make P_c larger. A larger value for P_c means a weaker strength of authorship proof, so this can never be used to improve the supposed strength of our watermark. This allows us to estimate p when it's exact value is not known.

The tactics follow below. For each tactic we discuss the type of constraints, the technique by which they are enforced, and the method of computing p, the probability of a constraint being satisfied purely by coincidence.

- **Merge random pairs of vertices.** Random vertices v_1 and v_2 are selected. If they are in the same partition then the constraint is considered satisfied. Otherwise it is broken. The merging process yields a graph that has both weighted vertices and edges. Each constraint is satisfied by coincidence in a k-way partitioning solution with probability $p = \frac{1}{k}$.
- **Add edges between random pairs of vertices.** Random vertices v_1 and v_2 are selected. The constraint is satisfied only if the two vertices are in the same partition. To make this more likely to occur, an edge is added between the two vertices. If there already is an edge between them, it's weight is increased by one. As above, the constraints are satisfied in a k-way partition by coincidence with probability $p = \frac{1}{k}$.
- **Merge random edges.** Choose a random edge e. The constraint is considered satisfied only if all of the vertices that are incident to the edge are in the same partition. The constraint is imposed by merging all of the vertices together. Let E be the number of edges in the original graph. Let $c(S)$ be the edge-cut of a particular partitioning solution S. Each constraint is satisfied by coincidence in a particular partitioning solution S with probability $p = \frac{(e-c(S))}{e}$.
- **Thicken random edges.** Choose a random edge e. The constraint is satisfied if all of it's terminals are in the same partition. This tactic makes this more likely to occur by adding one to weight of the edge. We refer to this as "thickening" the edge. As discussed above, the probability of a constraint being satisfied by coincidence is $p = \frac{(e-c(S))}{e}$.
- **Drop random edges.** Choose a random edge e. The constraint is considered satisfied if the edge is cut in the partitioning solution. This tactic removes the edge from the graph, so that constraints are more likely to be satisfied. Let E be the number of edges in the original graph and let $c(S)$ be the edge-cut of a particular partitioning solution S. Each constraint is satisfied by coincidence in a particular solution S with probability $p = \frac{c(S)}{e}$.

5. EXPERIMENTAL RESULTS

We report watermarking results on several benchmarks from the UCLA CAD Benchmarking Laboratory. The benchmarks we report on in this paper are shown in Figure 6. For our tests, we consistently overestimated p for the merge random edges, thicken random edges, and drop random edges tactics. Rather than use $c(S)$ to compute p we used the conservative lower and upper bounds given in the table.

The underlying partitioner we use is a circuit partitioner by Alpert, et al [2]. We experimented with both 2-way and 4-way partitioning [16]. We compared five different watermarking tactics which were described in the previous section.

	Description			2-way		4-way	
Graph	# Cells	# Nets	# Pins	low	high	low	high
19ks	2844	3282	10547	100	200	500	650
s13207	8772	8651	20606	50	120	400	700
s38584	20995	20717	55203	40	100	900	1400

Figure 6: Benchmark characteristics. "Low" values are used as conservative guidlines for determining P_c for the "merge hyperedges" and "thicken hyperedges" tactics. "High" values are used similarly for the "drop hyperedges" tactic.

Figure 7 shows the results for 2-way partitioning. The x-axis represents the edge-cut of the watermarked solution. The y-axis (scaled logarithmically) represents the probability P_c of achieving a solution by coincidence. There are five curves; one for each tactic. P_c is computed by a binomial formula as described in the previous section. The binomial formula takes the values C, b, and p as inputs. The value p can be computed in an obvious manner.

The values of C and b are not directly available from the figures, however. As an example, though, consider the point located at about $(126,\ 3\cdot 10^{-9})$ on the first figure in Figure 7. This is a point in the middle of the "drop random hyperedges" line. This point corresponds to dropping 300 random hyperedges (possibly "dropping" the same hyperedge more then once) from the circuit and then partitioning the resultant hypergraph. When this is done, 47 of the constraints are satisfied and 253 are broken. That is (if each of the 300 hyperedges selected were different) 47 of the 155 hyperedges that were cut are from our 300! This is amazing when you consider that the expected value is around 14. Computing the binomial with $C = 300$, $b = 253$, and $p = \frac{200}{3282}$, we get $P_c = 3.310987\cdot 10^{-09}$. Any partitioning solution of edge-cut 200 or less will have at most this probability of coincidentally containing the watermark. If we had chosen to set $p = \frac{155}{3282}$ instead, then we would get $P_c = 7.260486\cdot 10^{-13}$, but would only offer protection of this strength or more to solutions whose edge-cut was less than or equal to our own.

Apparent in all of the circuit partitioning expermental results is the superiority of the fifth, "drop random edges" tactic. It displays a very linear pattern on these figures, usually with a slope quite close to -1. This tactic's superior performance stems from a favorable tradeoff between the cost in edge-cut that is paid with each added constraint and the payoff in the strength of authorship proof that is gained with each added constraint.

6. CONCLUSION

Partitioning is an ubiquitous task in all synthesis and verification steps of the design process. We proposed the first approach for intellectual property protection of partitioning solutions using a watermarking scheme. Solutions produced using our approach simultaneously are very close to the best known solutions, carry signatures that are exceptionally unambiguous, are extremely unlikely to be present by coincidence, and are difficult to detect or remove.

7. REFERENCES

[1] C. J. Alpert, A. B. Kahng. Recent directions in netlist partitioning: a survey. INTEGRATION, the VLSI Journal, pp.1-81, 1995.

[2] C. J. Alpert, J.-H. Huang, A. B. Kahng. Multilevel Circuit Partitioning. 34th ACM/IEEE Design Automation Conference, pp.530-533, 1997.

[3] W. Bender, D. Gruhl, N. Morimoto, A. Lu. Techniques for data hiding. IBM Systems Journal, vol.35, no.3-4, pp.313-336, 1996.

[4] D. Chaum, H. vanAntwerpen. Undeniable signatures. Advances in Cryptography - CRYPTO '89 Proceedings, pp.212-216, 1989.

[5] D. Chaum, E. van Heyst. Group Signatures. Advances in Cryptography - EUROCRYPT '91, pp.257-265, 1991.

[6] L. Chen and T. P. Pedersen. New group signature schemes (cryptography). Advances in Cryptology - EUROCRYPT '94, pp.171-81, 1995.

[7] W. Diffie, M. E. Hellman. New directions in cryptography. IEEE Transactions on Information Theory, IT-22, pp.644-654, 1976.

[8] M. R. Garey, D. S. Johnson. Computers and intractability: a guide to the theory of NP-completeness. W. H. Freeman, 1979.

[9] I. Hong, M. Potkonjak. Techniques for Intellectual Property Protection of DSP Designs. ICASSP, pp.3133-3136, 1998.

[10] A. Kahng, et al. Watermarking techniques for intellectual property protection. Design Automation Conference, pp. 776-781, 1998.

[11] D. Kirovski, David Liu, Jennifer Wong, and M. Potkonjak. Forensic Engineering Techniques for VLSI CAD Tools. ACM-IEEE Design Automation Conference, 2000.

[12] A. J. Menezes. Handbook of applied cryptography. 1997.

[13] A.L. Oliviera. Robust Techniques For Watermarking Sequential Circuit Designs. Design Automation Conference, pp.837-842, 1999.

[14] I.Torunoglu, E. Charbon. Watermarking-based copyright protection of sequential functions. IEEE Custom Integrated Circuits Conference, pp.35-38, 1999.

[15] A.H. Tewfik, M. Swanson. Data hiding for multimedia personalization, interaction, and protection. IEEE Signal Processing Magazine, vol.14, no.4, pp.41-44, 1997.

[16] G. Wolfe, J. L. Wong, M. Potkonjak. Watermarking of Graph Partitioning Solutions. UCLA Technical Report 010002. March 2001.

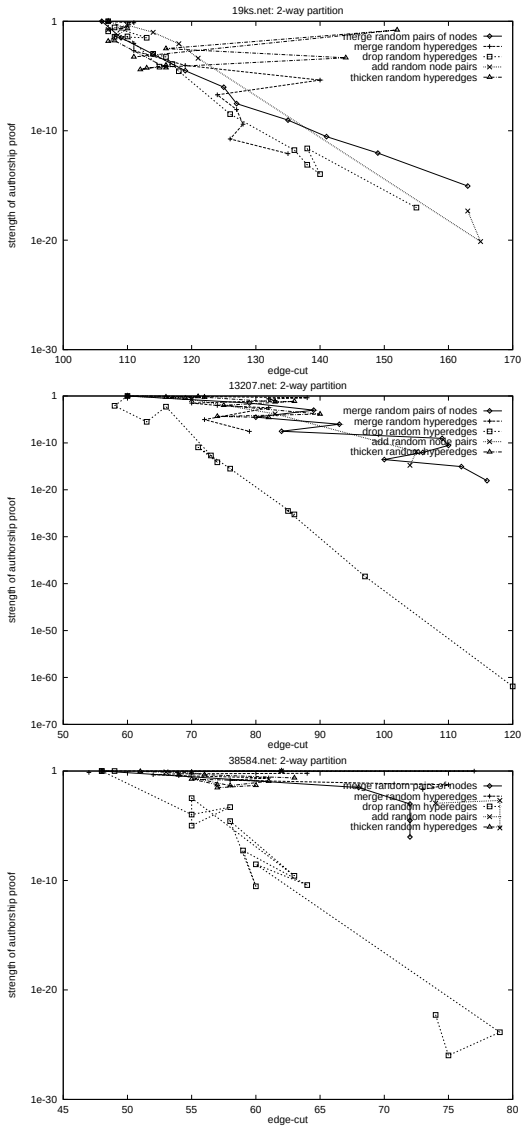

Figure 7: Trade off between degree of edge-cut degradation and strength of authorship proof for 2-way partitioning. Five tactics are compared.