Intellectual Property Metering

Farinaz Koushanfar[1], Gang Qu[2], Miodrag Potkonjak[3]
[1] EECS Dept., UC Berkeley, Berkeley, CA 94720 {farinaz@eecs.berkeley.edu}
[2] ECE Dept., University of Maryland, College Park, MD 20742 {gangqu@eng.umd.edu}
[3] CS Dept., UCLA, Los Angeles, CA 90095 {miodrag@cs.ucla.edu}

Abstract. We have developed the first hardware and software (intellectual property) metering scheme that enables reliable low overhead proofs for the number of manufactured parts and copied programs. The key idea is to make each design slightly different during postprocessing phase. Therefore, if two identical hardware/software designs or a design that is not reported by the foundry are detected, the design house has proof of misconduct.
We start by establishing implementation requirements for hardware metering. We also establish the connection between the requirements for hardware and software metering and synthesis process. Furthermore, we present mathematical analysis of statistical accuracy of the proposed hardware and software metering schemes. The effectiveness of the metering scheme is demonstrated on a number of designs and programs.

1 Introduction

1.1 Motivation, Key Idea, and Objectives

Our main goal in this paper is to introduce the first technique for hardware and software metering. The importance of these techniques is paramount for hardware and software intellectual property (IP) protection. It is estimated that more than $5B is lost annually to illegal manufacturing of integrated circuits. The number is significantly higher for illegal software reproduction related losses. These numbers are bound to increase rapidly, in particular for the hardware segment.

Hardware, design and semiconductor companies have been historically vertically integrated. Companies like IBM, Intel and NEC have both leading edge designs as well as superior foundry facilities. However, in the last five years there have been dramatic changes. The most profitable and fastest growing semiconductor business models have been in horizontally focused companies. On one side, pure contract silicon foundries, such as TSMC, UMC, and Chartered Semiconductor conquered almost 1/3 of all semiconductor world-wide output. On the other side, fabless design houses, such as Xilinx, Altera, Broadcom, and Juniper have been by far the fastest growing companies. There is wide consensus that in the future the horizontally focussed companies will significantly increase their market share.

One of major obstacles in this business model is that design companies do not have control over how many copies of their design are made by silicon foundries. Furthermore, FPGA companies get a significant part of their revenues by selling IPs which can readily be used on any of their chips without paying proper royalties. The case is even more crucial for software: once the user has the program, the only guarantee for the distributor that the users would not copy the programs are hardware/software locks and license agreements. It is much harder to find the illegal distributor of software since the reproduction sources are not limited. It is of utmost importance for the IP provider to meter the users of its programs. A number of companies consider development of hardware or intellectual property metering crucial for their business [37]. VSIA (Virtual Socket Initiative Alliance)

also identified hardware metering as one the key requirements for intellectual property protection. We propose a new intellectual property (IP) usage metering approach which allows IP providers to control proper collection of their IP royalties. The key idea of the hardware metering scheme is to make a very small part of the design programmable at the configuration time and to consequently configure this part for each manufactured chip in a unique way. Different configurations correspond to implementations which are differently scheduled or have different register assignments. Of course, this principle can be applied to other synthesis steps, including ones during logic synthesis or physical design.

Once when each manufactured chip or released software has a unique ID, it is relatively straightforward to enforce proper royalty agreements. For example, in hardware metering, if a foundry produces n chips which IDs are not reported to the design house in addition to p chips which are reported and approved, the probability that a randomly selected chip from the field has a non-approved ID is equal to n/n+p. Therefore with relatively few tests one can expect a high probability of detecting unauthorized chips.

An obvious, albeit naive, alternative to the proposed metering scheme is to just add a disconnected extra piece of programmable memories which carries the ID mark of a specific manufactured IC or to add extra identification code to the software. The first advantage of the proposed distributed and integrated within design hardware metering scheme over this straightforward scheme is that it has lower hardware overhead, since it leverages a part of don't-care signals in the finite state machine of the hardware design or an unused state in the software program. However, since the overall overhead for both schemes is low, there is a number of much more important advantages. What is common to all these attacks is that they externally induced controllability or observability. The approach also provides some level of protection against reverse engineering. For example in hardware**,** the presence of programmable control path instead of hard-wired logic makes reverse engineering more difficult since essentially all reverse engineering schemes require multiple chips to be dissected [1, 24]. Since, now each chip is slightly different but has the same functionality, the reverse engineering process is more difficult.

Furthermore, distributed programmable resources in the control part have a number of potential positive side effects. For example, they can be used to facilitate debugging [31] and engineering change during the design phase or testing once the chip is manufactured [10].

Finally, it is interesting and important to discuss the relationship of the proposed hardware metering scheme with fingerprinting schemes for IP protection [5]. For example, fingerprinting-based metering solution is to give the manufacturer the number of IPs as stated in the licensing agreement, each IP has a unique fingerprint and implements the same functionality [20]. If the manufacturer uses one piece of IP more than once, then they face risk of being caught by the IP provider from detecting multiple copies of the same fingerprint. However, this challenges the mass foundry production line since each IP requires the unique mask and makes tuning of parameters of the foundry line to design much more difficult. Also, fingerprinting will inevitably introduce a significantly large overhead since it aims at placing hidden information in all parts of the hardware/software design and follows random signature driven constraints.

1.2 Motivational Example

To illustrate the key ideas behind the hardware metering approach, consider the second order continued fraction IIR filter [9] shown in Figure 1. For simplicity, we assume

that all operations take one control step. The critical path is six control steps long. We schedule the filter, in the minimum amount of hardware resources using only 1 adder and 1 multiplier. The graph in Figure 2 shows the same filter after being restructured following the schedule. The same graph also has information about the used variables which are denoted by (v1, v2,..., v11).

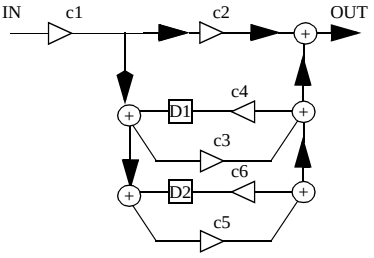

Figure 1. 2nd order continued fraction IIR [CFIIR] filter

A variable is alive during its lifetime, i.e. between the time it is generated (written) and the last use (read) of it. The variables whose lifetime do not overlap can be stored in the same register. Figure 3 shows the interval graph that contains information about lifetimes of all variables for the filter. The standard way of variable assignment to registers is to model it using the graph coloring problem [6, 36]. The interval graph is constructed in such a way that for each variable, there is a corresponding node in the graph. Two nodes are connected if the lifetime of the corresponding variables overlap. Now, register assignment can be performed by coloring the interval graph, which is an NP-complete task for cyclic interval graphs [16,21].

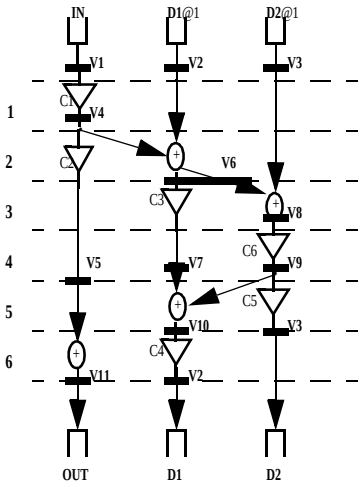

Figure 2. Scheduled CFIIR filter

The instance of the graph-coloring problem that corresponds to register assignment of the filter is shown in Figure 6 (considering only the solid lines). Assigning two variables to the same registers corresponds to coloring two nodes with the same color. One potential assignment is also shown in Figure 6. Finally, Figures 4 and 5 show the corresponding datapath and a path of control unit (FSM) that contains read/write operations to the register. Figure 5 shows read control of the register files that is used to store variables. The key point is that although we can obtain many different solutions (which we discuss in the next paragraphs) by coloring the graph in different ways, our datapath remains the same for

every possible solution which uses the minimal number of colors. The only difference is in terms of the control unit (FSM). Since the datapath is not modified, we can use the same mask for all layouts. The control unit is a very small fraction of the total layout area and we implement it by using a programmable technology such as EEPROM. The most appealing advantage of EEPROM is that it is not programmed during the mask steps, but can be later configured in a fast and cost effective way.

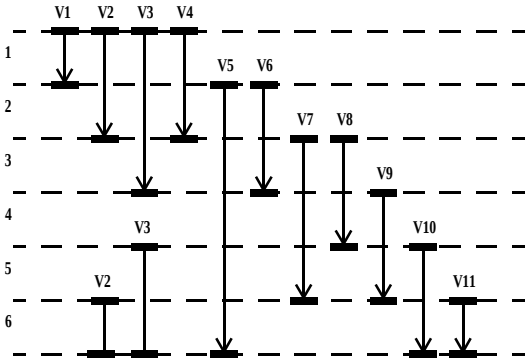

Figure 3. The interval graph of the CFIIR filter

There are several ways how to produce solutions with identical datapaths and distinct control paths. For example, one option is to permute the subsets of variables that are assigned to particular register. There are *n!* solutions, when *n* registers are used in the register file for a set of variables. Another alternative is that by using degrees of freedom in assigning the variables to registers or by using redundant states for the same FSM and adding new constraints to find different control flows during the logic synthesis phase.

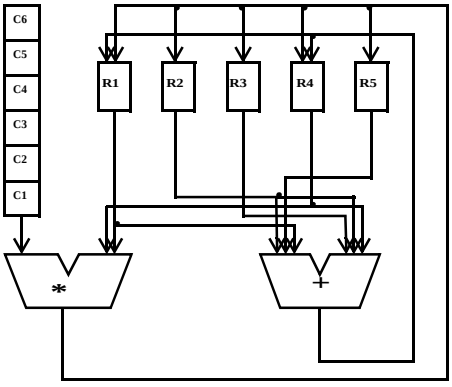

Figure 4. Datapath for the CFIIR filter

We illustrate the first option by adding dashed-lines on the graph in Figure 6. The example is the constructed from Figure 6 by adding one extra node, u1. The node v1 is connected to 3 other nodes (this is shown by the dash-lines on the figure). Since we are using 5 colors to color our graph (since 5 variables are simultaneously alive in control step 5 that is an minimal solution), we have a degree of freedom of 2 to color this node. In the previous section, we have used R1 to color this node, but now we see that we can also use R5 to color it. Note that the same effect can be done as preprocessing step as explained in Section 5.

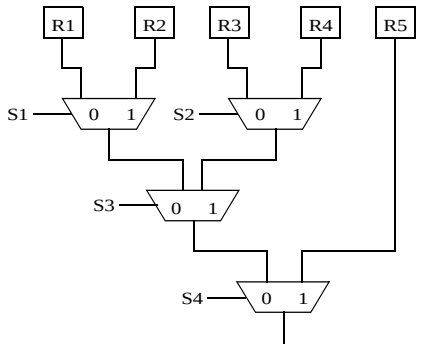

Figure 5. Programmable read logic for registers

During logic synthesis of control path we can create for some states redundant equivalent states. Each of variables will obtain different state assignments, and we can use any of the assignments of the two equivalent states. If these step is repeated n times, we will have 2^n different solution. The remainder of the paper is organized in the following way:

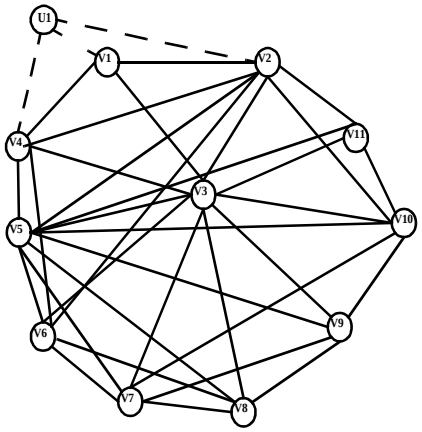

Figure 6. Graph-coloring interpretation of the register assignment problem. The variable assignment is the following: v1-R1; v2-R2; v3-R3; v4-R4; v5-R1; v6-R5; v7-R2; v8-R4; v9-R5; v10-R4; v11-R5

We first survey the related literature. Next we formulate the metering problem by establishing objectives and figures of merit. After that, we propose and analyze a number of hardware and software metering techniques. Probabilistic analysis of the metering technique and explanation of technological and implementation issues is given in Section 5. After that we present our synthesis and optimization approach and algorithm. Finally, we present experimental results and conclude by briefly discussing future potential metering directions.

2 Related Work

To the best of our knowledge this is the first approach for hardware IP metering. Related work can be traced within broad fields of cryptography and computational security, conceptually related fields of intellectual property protection and licensing and objective-related field of software, content, and WWW access metering. Recently, SiidTech

Inc., an Oregon start-up company, has proposed an approach for integrated circuit identification from random threshold mismatches in an array of addressable MOSFETs. The technique leverages on process discrepancies unavoidably formed during fabrication. This analog technique can be used in tracking semiconductor dies, authentication and intellectual property (IP) tagging. In a recent report of this method's measured performance[23], for a 0.35um poly CMOS, for generating 112 ID bits, 132 blocks area used, each with the area of 252x93um. The IDs proposed by SiidTech are not deterministic and these IDs can not be deterministically compacted. Also, due to the birthday paradox, there is still a small probability that two IDs generated randomly have the same value. Component applications enables the user to trace a particular die on a wafer and store this information for future usages. There are several advantages of our scheme over the Siid scheme. We have been able to obtain more than 1.3E12 distinct solutions even in our smallest test cases which have only 15 registers (Our number of solutions will go exponentially high by using a few more registers). Furthermore, our IDs are deterministic and therefore they can be used to contain a defined signature to be used in many cryptographic schemes.

From a more global point of view, intellectual property protection schemes, such as hardware metering, are traditionally treated within computational security. Cryptography is the study of techniques related to aspects of information security such as confidentiality, data integrity and entity authentication [34]. Computational security has even more broad scope and includes privacy protection, password protection, denial of service, and content usage measuring.

Modern cryptography started with introduction of one-way trapdoor function-based public-key cryptographical communication protocols by Diffie and Hellman [11]. In 1978, Rivest, Shamir and Adleman discovered the first practical sound public key encryption and signature scheme based on exponential computational difficulty of factoring numbers which are the products of two large prime numbers. A number of excellent cryptographical textbooks are available [35, 34, 25].

Intellectual property protection of audio and video artifacts and hardware and software components and systems recently attracted a great deal of attention. For example, the Virtual Socket Interface Alliance has been making progress on standardizing SoC design and IP protection [38].

Multimedia watermarking schemes are utilizing minute alternations of audio or video so that the signature is embedded while human perceived quality of artifact is fully preserved. For survey, see [18], Protecting design (hardware and software) IP is a broad and complex issue.

One method to enable design IP protection is based on the constraint manipulation. The basic idea is to impose additional author-specific constraints on the original IP specification during its creation and/or synthesis. The copyright detector checks whether a synthesized IP block satisfies the author-specific constraints. The strength of the proof of authorship is proportional to the likelihood that an arbitrary synthesis tool incidentally satisfies all the added constraints [19, 32]. Similarly, to protect legal users of the IP, fingerprints are added to the IP as extra constraints [5]. Finally, copy detection techniques for VLSI CAD applications have been developed to find and prove improper use of the design IP [7,20]. These techniques are effective for authentication. However, since they make each design unique, it becomes ill-suited for mass-production and cannot be applied, at least not directly, and without significant modification, to hardware metering.

Another research, to some extent related to our work, is forensic engineering technique, that has been explored for detection of authentic Java byte-codes [3] and to perform identity or partial copy detection for digital libraries [4]. Also, forensic analysis principles are used in the VLSI CAD to demonstrate that solutions produced by strategically different algorithms can be associated with their corresponding algorithms with high accuracy [22].

Sampling and auditing are the two main methods for measuring the popularity of media channels. Sampling, like the Nielsen Media Research and NetRatings Inc., is based on surveys among a representing group of users [30]. Web page access metering has been addressed by a number of researchers and companies. For example, Pitkow proposed techniques to uniquely identify users and to compensate for the usage of proxies and cashes [30]. Franklin and Malkhi developed the lightweight security WWW page usage scheme [15]. Recently, Naor and Pinkas proposed a rigorous secret sharing-based WWW page access method [26]. Another potential alternative is to use micro-payment protocols for WWW usage [27].

Majority of software vendors currently employ licensing as the primary way of protecting their software packages, such as text formatting and CAD tools. Licensing software ensures the vendor with a certain degree of control over the distributed software. For example, licensing software may prevent unauthorized duplication of software packages or Licensing is a major enabling component for software distribution. For example, over $40 billion of installed third party software uses GLOBEtrotter's Electronic commerce for software technology FLEX1m. Today's dominating software licensing mechanism is based on license key concept. A key is encrypted by using a string of data that contains software package ID and its usage constraints (e.g. expiration date) and the serial number of the computer where the key is installed. The invocation of the software package is done automatically when software is invoked by using one of the password schemes [25, 13].

A large number of patented licensing protocols have been proposed. For example, licenses can be used not only to authenticate the legal users, but also to upgrade the products, and other after-market information transmissions[28] or licensing using smart cards [29, 2].

3 Preliminaries

3.1 Problem Scenario

Consider the following scenario that requires hardware metering: a start-up design company *A* builds a system that outperforms all the similar products on the market. *A* gives the VHDL description of the system to manufacturer *B* and makes an agreement with *B* to fabricate 10 million copies. The first 2 million copies sold out almost immediately, then the sale slows down even when company *A* lowers the price. It seems the market has already been saturated. Meanwhile, market survey shows that there are about 12 million similar products in use. *A* suspects that foundry *B* has violated the agreement and fabricated more than 10 million copies without reporting to *A*. However, for a given product, *A* cannot provide convincing evidence to tell whether this copy is legally sold or not. Therefore, *A* fails to recover its R&D revenue.

We observe that the problem comes from the fact that *A* sells identical products on the market. If they can give each product a unique identification number, then when two products with the same identification number are found, the existence of unauthorized

becomes obvious. One naive approach is to use a serial number, however, it is visible and almost trivial to be removed. In this paper, we propose a scheme that embeds a unique identification number inside of the product.

3.2 Requirements and Objectives

Before the discussion of technical details, we first analyze the requirements and objectives. Four basic questions have to be answered:

P1 How to create many distinct copies with the same functionality?

P2 Once two identical copies are found, how can we prove our ownership, i.e., how can we convince others that we are not the pirates?

P3 How many tests we need to conduct before we gain a certain level of confidence that there are no unauthorized on the market?

P4 If there are unauthorized copies, how can we estimate the number of copies that they have made?

The existing watermarking techniques provide solutions to problem **P2**: During the design synthesis, we embed our digital watermarks and later on retrieve such watermarks for authorship [19]. The last two questions are interesting for obvious reasons. **P3** estimates designer's effort to prove foundry's honesty, while **P4** provides valuable on-court information for the designer. We will build statistical models and address them in the next section. To end this section, we discuss the requirements for solutions to the first question:

- Correct functionality: Although we want the system to be distinct, they must have exactly the same functionality.
- Large number of different copies: The method has to be capable of producing huge amount of distinct copies (from tens of thousands to millions) to accommodate to the market.
- Low overhead: The degradation of system's performance due to the large number of different copies has to be kept at the minimal level, if zero-overhead is not achievable.
- Transparent to design and test: The creation of different copies has to be transparent to the manufacturing and testing. Otherwise, it will make the mass production impossible. For this reason, we suggest post-processing, i.e. keep most components of the chip the same and make small changes at the final stage of the fabrication.
- Resilient against attacks: Attempts to making distinct extra copies or duplicated copies without being caught will be difficult, costly, and time-consuming.

4 Hardware and Software Metering Techniques

In this section, we propose and analyze a number of ways for hardware and software metering. There are several alternatives for implementing the identification logic within the control path logic for hardware protection. Our focus is control logic, because in modern design it is usually a very small percentage of design area, often less than 1%. Each of the proposed techniques which have certain advantages/disadvantages.

4.1 Sequential Memory-Based Approach

In this approach the required data is stored in a family of PROMs (preferably non-reconfigurable e.g. OTP EPROMs). This data is then read out of the registers sequentially to form a control path. The fast improving memory technology is rapidly reducing on-board programming time and the required extra manufacturing processing steps. The advantages of this approach includes on-board programmability and small area overhead. However,

the additional required mask steps and erasure of UV light for programming the PROM, somewhat limits attractiveness of this approach.

4.2 Disconnection Approach

In this approach, an additional finite state machine (FSM) is designed to facilitate design identification. Checking the ID of the design, requires an unused state of the other FSMs that are part of the design. Modern designs have a large number of FSM with numerous unused states/input combinations (don't cares). The added FSM, is the same for all the designs in the mask level. In the postprocessing step, lasers burn some of the connections of this added FSM in each design and thus generates different states and functions of it. This added FSM is different in each design since we laser burn different connections in each design to achieve a slightly different control path. The algorithms to decide exactly where to burn the interconnect in each chip, can be derived from a computer simulation of the state machine to derive unique ID for each of them. This solution does not need any extra processing steps and is much faster and more robust than the previous approaches. Another alternative, is to use BISR mechanism for hardware metering. BISR designs are designs which have built in self repair fault tolerance that can function properly even if some parts of the design are faulty [17]. The idea here is to intentionally induce variety of faults in BISR designs in such a way that each design has different faulty parts. Note that while repairing circuits using BISR is relatively expensive, inducing faults is relatively cheap.

4.3 Fingerprinting by using the SiidTech Approach

This solution uses the same methodology as the disconnection approach mentioned in the last section. The difference is that the added FSM is now reading out the unique fingerprint proposed by the SiidTech Corporation [23].

SiidTech approach, which identifies each chip by detecting imbalances in threshold voltages-discrepancies unavoidably formed during fabrication. The advantage of this approach is also that no external programming or special processing steps are needed. A silicon fingerprint is generated at "birth" - during the fabrication of the die - and is carried throughout the silicon's useful life. The disadvantages of this approach are the same as for the generic Siid technology that was elaborated in Section 2.

4.4 Software Metering

The proposed hardware metering techniques are directly applicable to software metering. Actually, it is significantly easier to create software tracking techniques, since there is no technological constraints associated with integrated circuit manufacturing process. One alternative is to just add static variants in software executables, by imposing constraints during compilation. The constraint can be either local or global.

Another alternative for hardware and software metering is to use dynamic data structures as ID carriers. This approach has been already used for software protection through watermarking [8]. The authors propose a combination of code obfuscation with creation of multiple versions for a set of dynamic data structures created during the program execution.

The idea of dynamic creation of ID can be applied for software metering. We propose radically different approach to this task. We add an extra software module which is the ID carrier. For each copy this module is differently configured. The module is invoked from

the main flow of the program when a special function is invoked from command or highly unlikely data input. Our preferred mode is to use don't care conditions in the control flow of the program to enter the ID module. Program obfuscation can be used to protect the module isolation and altering against attacks. The module may have either dynamic data structures as information carrier or can just create a particular output sequence.

In addition to mechanism differences, there two other key differences between the schemes discussed in [8] and the one just proposed. The first is that our goal is different: instead of watermarking, we want only creation of distinct copies. This condition makes not only the software mutation task easier, but also induces low size and/or performance overhead. The second difference is even more important. Our primary technical goal is to enable rapid ID authentication for a software component/product. Therefore, the idea is to create versions that rapidly create different IDs that can be verified against database of produced copies. Note that our technique can be used, with straightforward modification, also for software watermarking. The final alternative to software metering which we discuss is conceptually most complex, but also potentially most rewarding. The idea is to alter software in such a way that each output produces by different version of the software which differs from each other. This can be easily achieved by postprocessing the final output within the software. More interestingly, it can be demonstrated that in many cases one can systematically alter software in such a way that the functionality is essentially preserved. Typical example include word processing, large scale optimization and computer-aided design packages.

5 Detection: Mathematical Model and Results

In this section, we will address problems **P3** and **P4** proposed in Section 3. Suppose the design house asks the foundry to fabricate n copies and N Š n is the number that the foundry really makes. **P3** asks the expected number of tests to find a duplicate if $N > n$ or the number of tests to convince designer that $N = n$. **P4** requires an estimation of N once the first unauthorized is found. We take the dishonest foundry's best strategy in that he makes $k-1$ duplicates for each original copy. It is proven that for a fixed $N = k \cdot n$, the dishonest foundry has the best chance to survive in this equiprobable case.

Theorem 5.1. Draw l from $N = k \cdot n$ objects which consist of k copies of n distinct ones, the probability that there is no duplicate, denoted by *Prob[n,k,l]*, is

$$\left[1-\frac{k-1}{N-1}\right]\cdot\left[1-\frac{2(k-1)}{N-2}\right]\cdots\left[1-\frac{(l-1)(k-1)}{N-(l-1)}\right] \tag{1}$$

which has an upper bound

$$\left[1-\frac{p}{n}\right]\cdot\left[1-\frac{2\cdot p}{n}\right]\cdots\left[1-\frac{(l-1)\cdot p}{n}\right] \tag{2}$$

where $p = 1-1/k$.

Prob[n,k,l] is the probability that there are no unauthorized parts found after l random tests (without replacement), provided that there are k copies for each of the n originals. It decreases as k increases, since when the population (N) grows, it becomes more difficult to find duplicates; it also decreases as l, the number of tests, increases.

The quantity 1-*Prob[n,k,l]* is the confidence that the designer can achieve from l consecutive successful tests. Success means that no duplicate is found. Table 1 shows some

examples for the case n=1000. For instance, after checking 50 products and not finding any duplicates, the designer believes that there does not exist another copy of 1000 chips with a 46.64% confidence. With the same result, the probability that the foundry makes 10000 instead of 1000 is less than 33% (1-67.37%). The designer's confidence goes up quickly as more tests are conducted. After 100 successful tests, the designer will be 92.62% convinced of the foundry's honesty.

Table 1: Designer's confidence after l consecutive successful tests

l	k=2	k=3	k=4	k=5	k=10
10	2.24%	2.97%	3.33%	3.55%	3.98%
20	9.15%	12.00%	13.38%	14.20%	15.82%
50	46.64%	56.62%	60.87%	63.21%	67.47%
75	76.34%	85.25%	88.33%	89.86%	92.33%
100	92.62%	96.84%	97.73%	98.39%	99.02%

One implication of Theorem 5.1 is the "Birthday Paradox" problem: among 24 people, with probability larger than one half, there will be two who share the same birthday, assuming all birth dates are equally distributed over the days in the year.

Theorem 5.1 not only gives formula on the designer's confidence about foundry's honesty, it also answers problem **P3**. As we mentioned, 1-*Prob[n,k,l]* measures the foundry's honesty and it increases as l increases. For a designer to gain a desired level of confidence α, we need to find the smallest l such that $1 - Prob([n, k, l] \geq \alpha)$. Unfortunately, there is no exact closed form for formula (1), however, the solution can be always found numerically and there exist good approximation formulas when n is large [14].

We assume that k is equally distributed and derive Theorem 5.2 which answers problem **P4** immediately.

Theorem 5.2. The probability that the first unauthorized is found at the l+1st test is

$$Pr[n, k, l+1] = Prob[n, k, l] \cdot \frac{l \cdot (l+1) \cdot (k-1)}{N - l} \tag{3}$$

Corollary 5.3. The expected number of tests to find the first unauthorized is

$$\sum_{k=1}^{\infty} \sum_{l=1}^{n(k-1)+1} l \cdot Pr[n, k, l] \tag{4}$$

Corollary 5.4. If the first failure occurs at l, then the expectation for k is

$$\sum_{k=1}^{\infty} k \cdot Pr[n, k, l] \tag{5}$$

6 Global Design Flow

In this section, we address how to create many different copies of the systems that have the same functionality.[1] We illustrate our approach using two problem instances that are heavily used in the VLSI CAD, namely: graph coloring, and boolean satisfiability.

1.

6.1 Graph Coloring

The NP-hard graph vertex coloring (GC) optimization seeks to color a given graph with as few colors as possible, such that no two adjacent vertices receive the same color.

Given a graph, our objective is to create as many as possible high quality solutions that are relatively close. By high quality, we mean that if the optimal solution is known, then all the solutions that we generate will not use any extra color. Therefore, the fingerprinting techniques for GC cannot be used in this case, because they usually introduce overhead although they are very effective in creating new solutions.

The following steps illustrate our algorithm for GC solution generation.

1. Apply a graph coloring heuristic to color the given graph $G(V, E)$ and obtain a *k*-color scheme as the seed solution.
2. For each node $v \in V$, calculate *c(v)*, the number of different colors that *v*'s neighbors get.
3. Sort the nodes *V* in the increasing order of *c(v)*.
4. For each node $v \in V$ with $c(v) < k-1$, change *v*'s color and report $k-1-c(v)$ different solutions.
5. For all pairs of nodes *(u,v)* with $c(u) < k-1$ and $c(v) < k-1$, try different coloring schemes for nodes *u* and *v* and report the new found solutions if any.

In next section, we will demonstrate the performance of this algorithm by experimental results. It turns out that this simple strategy works very well in real-life graphs. Notice that no extra colors will be used in our approach, i.e., all the derived solutions will have the same quality as the seed solution. And these solutions differ from the seed solution only at the colors of one or two nodes.

6.2 SAT

The boolean satisfiability problem (**SAT**) seeks to decide, for a given formula, whether there is a truth assignment for its variables that makes the formula true. We necessarily assume that the SAT instance is satisfiable and that there is a large enough solution space to accommodate multiple solutions.

We use pre-processing techniques to create different but close solutions for the SAT problems. In particular, before we solve the SAT instance, we delete a selective subset of variables and essentially make them "don't-cares". Suppose we introduce *k* such "don't-cares", then we should be able to build 2^k distinct solutions from one seed solution if it exists. Moreover, these 2^k solutions will assign exactly the same value to the variables that are not selected, i.e., they are close.

We select the variables to be deleted iteratively and greedily based on the following criteria: for each variable *v*, let n_v be the number of clauses that contains either *v* or v', and let s_i be the length of the i^{th} such clause. Define

$$c(v) = \sum_{i=1}^{n_l} \frac{1}{2^{S_i - 1} - 1} \tag{6}$$

1. For each variable *v* in formula *F*, calculate *c(v)* and unmark *v*.
2. Select a unmarked *v* with the smallest *c(v)*, delete both *v* and *v'* from *F* to create a new formula *F'*.
3. Apply a SAT solver to solve *F'*.

4. if (F' is satisfiable)
   $F = F'$ and goto step 1.
5. else
   mark v and goto step 2.

We select variable v greedily in step 2 and modify the formula in step 3. We use SAT solver to solve the new SAT formula (the one without variable v), if we fail to find a truth assignment, we put v back and select the variable that has the second smallest $c(v)$. If the new formula is still satisfiable, then we recalculate $c(v)$ for the remainder variables in the modified formula F' and select the next one. We continue this process until we find enough "don't-cares" to create the desired number of different solutions.

7 Experimental Results

In this section we analyze the ability of the proposed metering scheme to generate a large pool of designs with unique ID. We first show our results for hardware metering. The last part of this section illustrates the analysis for the software metering approach.

Table 2 shows the results of the application of the scheme on generating numerous graph coloring (register assignment). The first column indicates the name of design from the Hyper benchmark suite [33]. The second and third column indicate the number of variables and registers in the designs. Two final columns indicate the number of the unique solutions which can be obtained using the following two methods. The first one (column 4) is the assignment of exactly the same subset of variables to different registers in their physical instances. The last column indicates the number of different solutions produced using the technique presented in Section 5. In both cases, even for the smallest design, the number of solutions is very high. The key reason for this situation is that it is well known that the interval graphs for all known designs are very sparse and it is very easy to color them in many different ways using the minimal number of colors.

In order to test the technique in a much more demanding scenario, we applied the hardware metering scheme on the SAT problem. The experimental results are shown in Table 3. The first column indicates the name of DIMACS benchmark [12] and the middle column indicates the number of used variables. The last column indicates the number of solutions that were generated using the technique presented in Section 4. Although, the number of the generated solutions is smaller than in the case of graph coloring, it is still very large and much higher than required in any of today's designs.

Table 2: Generated number of distinct solutions for the register assignment-based metering scheme

Design	Variables	Registers	#of solutions	
8th CD IIR	35	19	1.2E17	1.1E21
Linear GE Ctrl	48	23	2.6E22	5.0E36
Wavelet	31	20	2.4E18	9.4E17
Modem Filter	33	15	1.3E12	5.9E18
2nd Volterra	28	15	1.3E12	9.0E16
D/A Converter	354	171	> 1E200	5E123
Echo Canceler	1082	1061	> 1E200	6E202

Table 3: Generated number of distinct solutions for the SAT-based metering scheme

File	Variables	#of solutions
ii8a1	66	1.3E08
ii8a2	180	5.8E17
ii8a3	264	3.2E32
ii8a4	396	4.3E37
ii8b1	336	5.0E27
ii8b2	576	2.3E46
ii8b3	816	5.7E73
ii8b4	1068	5.1E89

8 Conclusion

We have developed the first hardware and software (intellectual property) usage metering scheme. The scheme enables design companies to securely control licensing rights for their IP. The utilizes a small percentage of a design implemented using configurable technology to embed a unique ID to each manufactured design instance. This scheme is generalized in a number of ways and applied to both hardware and software metering.

We also presented mathematical analysis for detection accuracy of the proposed scheme. We demonstrated the ability of the scheme to implement very high number of chips with different ID. The main result of the paper is that we established generic connection between the scheme and synthesis and compilation tasks.

References

[1] R. Anderson, M. Kuhn, "Tamper resistance-a cautionary note."USENIX Workshop on Electronic Commerce, pp. 1-11, 1996.

[2] T. Aura, D. Gollmann, D. "Software license management with smart cards. Proceedings of the USENIX Workshop on Smartcard Technology (Smartcard'99), pp.75-85, May1999.

[3] B.S. Baker and U. Manber. "Deducing similarities in Java sources from bytecodes," USENIX Technical Conference, pp.179-90, 1998.

[4] S. Brin, J. Davis, and H. Garcia-Molina. "Copy detection mechanisms for digital documents," SIGMOD Record, vol. 24, No. 2, pp. 398-409, 1995.

[5] A.E. Caldwell, H. Choi, A.B. Kahng, S. Mantik, M. Potkonjak, G. Qu, and J.L. Wong. "Effective iterative techniques for fingerprinting design IP," Design Automation Conference, pp. 843-848, 1999.

[6] G.J. Chaitin. "Register allocation and spilling via graph coloring." SIGPLAN '82 Symposium on Compiler Construction, pp. 98-105, 1982. Design Automation Conference Proceedings, pp. 843-848, June 1999.

[7] E. Charbon and I. Torunoglu. "Copyright protection of designs based on multi source IPs," IEEE/ACM International Conference on Computer Aided Design, pp. 591-595, June 1998.

[8] C.S. Collberg, "Reverse interpretation + mutation analysis = automatic retargeting." Proceedings of the ACM SIGPLAN 1997 Conference on Programming Language Design and Implementation, pp. 15-18, June 1997

[9] R.E.Crochiere, A.V. Oppenheim. "Analysis of linear digital networks." Proceedings of the IEEE, vol.63, (no.4):581-95., April 1975

[10] S. Dey, V. Gangaram, M. Potkonjak, "A controller redesign technique to enhance testability of controller-data path circuits." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol.17, (no.2):157-68. Feb 1998.

[11] W. Diffie,M. Hellman, "New directions in cryptography." IEEE Transactions on Information Theory, vol.IT-22, (no.6) p.644-54, Nov 1976.

[12] http://www.dimacs.rutgers.edu

[13] R. Findley Jr., Raymond, R. Dixon, "Dual smart card access control electronic data storage and retrieval system and methods" US patent#5629508, May 13, 1997.

[14] P. Flajolet, D. Gardy, and L. Thimonier. "Birthday paradox, coupon collectors, caching algorithms and self-organizing search." Discrete Applied Mathematics, Vol. 39, No. 3, 11 pp. 207-229, November 1992.

[15] M.K. Franklin, D. Malkhi, "Auditable metering with lightweight security". Journal of Computer Security, vol.6, (no.4), IOS Press, 1998. p.237-55.
[16] M.R. Garey, D.S. Johnson. "Computers and intractability. A guide to the theory of NP-completeness." Oxford, UK: Freeman, 1979.
[17] L.M. Guerra, M. Potkonjak, J.M. Rabaey, "Behavioral-level synthesis of heterogeneous BISR reconfigurable ASIC's." IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.6, (no.1), March 1998. p.158-67.
[18] F. Hartung, M. Kutter. "Multimedia watermarking techniques." Proceedings of the IEEE, July 1999, vol.87, (no.7):1079-107.
[19] A.B. Kahng, J. Lach, W.H. Mangione-Smith, S. Mantik, I.L. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe. "Watermarking techniques for intellectual property protection," 35th ACM/IEEE Design Automation Conference Proceedings, pp. 776-781, June 1998.
[20] A.B. Kahng, D. Kirovski, S. Mantik, M. Potkonjak, and J.L. Wong. "Copy detection for intellectual property protection of VLSI design," IEEE/ACM International Conference on Computer Aided Design, pp. 600-604, November 1999.
[21] D. Kirovski, M. Potkonjak. "Efficient coloring of a large spectrum of graphs." Proceedings 1998 Design and Automation Conference. 35th DAC. pp.427-32., 1998.
[22] D. Kirovski, D. Liu, J.L. Wong, and M. Potkonjak. "Forensic engineering techniques for VLSI CAD tools," 37th ACM/IEEE Design Automation Conference Proceedings, June 2000. pp.581-6.
[23] K. Lofstrom, W.R. Daasch, D. Taylor, "IC identification circuits using device mismatch" Proceedings of the International Solid-State Circuits Conference, pp. 372-3, 2000.
[24] D. P. Maher "Fault induction attacks, tamper resistance, and hostile reverse engineering in perspective.", Financial Cryptography First International Conference, pp. 109-21, 1997.
[25] A. J. Menezes, P.C. van Oorschot, S.A. Vanstone, "Handbook of applied cryptography", Boca Raton, FL:CRC Press, 1997.
[26] M. Naor, B. Pinkas, "Secure accounting and auditing on the Web". Computer Networks and ISDN Systems, vol.30, (no.1-7), Elsevier, April 1998. pp.541-50.
[27] R.L. Rivest (Edited by: Hirschfeld, R.), "Electronic lottery tickets as micropayments." Financial Cryptography First International Conference, pp.307-14, 1997.
[28] C.D. Ross, N.W. Taylor, K.W. Kingdon, H.R. Davis, D. Major "Method and apparatus for electronic licensing", US patent# 5553143, Sept. 3, 1996
[29] D.C. Thomas "Method and apparatus for providing security for computer software", US patent#4446519, May 1984
[30] J. Pitkow, "In search of reliable usage data on the WWW". Computer Networks and ISDN Systems, vol.29, (no.8-13), Sept. 1997. p.1343-55.
[31] M. Potkonjak, S. Dey, K. Wakabayashi, "Design-for-Debugging of application specific designs." International Conference on Computer-Aided Design, pp. 295-301, 1995.
[32] G. Qu and M. Potkonjak. "Analysis of watermarking techniques for graph coloring problem," IEEE/ACM International Conference on Computer Aided Design, pp. 190-193, November 1998.
[33] J.M. Rabaey, C. Chu, P. Hoang, M. Potkonjak, "Fast prototyping of datapath-intensive architectures." IEEE Design & Test of Computers, June 1991, vol.8, (no.2):40-51.
[34] B. Schneier, "Applied cryptography: protocols, algorithms, and source code in C /". 2nd ed. New York: Wiley, 1996.
[35] D.R. Stinson, "Cryptography: theory and practice". Boca Raton, FL: CRC Press, 1995.
[36] L. Stok, J.A.G. Jess, "Foreground memory management in data path synthesis." International Journal of Circuit Theory and Applications, vol.20, (no.3):235-55. May-June 1992
[37] K. Veenstra, Altera Corporation, Personal Communication., January 1999.
[38] http://www.vsia.com