Copyright Protection of Designs Based on Multi Source IPs

Edoardo Charbon and Ilhami Torunoglu
Cadence Design Systems, San Jose, CA 95134

Abstract

This paper addresses the copyright protection problem of integrated circuits designed with blocks which are originated from multiple design sources. The process consists of two phases. First, a compact signature is generated from every block independently and made public. Utilizing such signatures, a design can be decomposed into its original building blocks, regardless of multiple hierarchies. Then, a map of all the blocks can be built, thus allowing to reconstruct the original copyright dependencies. The proposed methodology can be used by foundries to verify that designs submitted for fabrication contain blocks traceable to a legal source of intellectual property. The verification process is also useful to intellectual property providers and integrators, as it reduces the likelihood of infringement, thus ultimately minimizing the risk of litigation.

1 Introduction

The revolution introduced by new paradigms based on massive design reuse and virtual block integration will touch most aspects of current design flows. One of the most important among emerging issues involves Intellectual Property (IP) infringement. IP providers and integrators will be significantly exposed to this form of industrial espionage. Nonetheless other sections of this new industry will be at a comparable, if not higher, risk. One of such industries is that of silicon foundries. The reason for such a high degree of liability exposure is due to the fact that foundries produce the first tangible proof of potential infringement embedded in fabricated chips.

Currently, integrators must provide the foundry with a complete description of all the virtual components present in a design prior to submitting it for fabrication. Copyright fees due to IP providers are generally paid for at the source, i.e. by the integrator. In some cases a design may consist of virtual components for which a royalty agreement was not settled. By agreeing to fabricate the design, a foundry is liable of infringement and could potentially incur in costly legal expenses to settle disputes with the IP provider. For these reasons, detecting and tracking espionage prior to fabrication could be a cheaper and far more efficient alternative. Furthermore such a detection method, if advertised, may prove to be a powerful deterrent to discourage potential theft.

This paper proposes a novel technique based on a two phase process. First, from the IP a string of symbols, or *signature*, is generated and made public. This phase is called *registration*. Later, when such IP will be integrated into a larger design, it will be possible to extract its signature, hence revealing the presence of the IP in the design. The process of extracting all embedded designs by means of their signatures is known as *detection*. The first phase should be implemented by all the IP providers who wish to protect their copyrights. The second phase should be a prerogative of foundries to ensure that no infringed circuits are present in any given tape-out. Integrators may also register IP signatures, so as to ensure protection of their IP contributions to the design being fabricated.

Figure 1 shows the approach with the registration and detection phases. Note that even though a signature can be generated from any given design, it does not contain sufficient information to allow one to reverse engineer the original design. Thus, the IP signature bank could be maintained by a third party so as to ensure impartiality in case of litigation. For each tape-out the foundry can construct a map of all the IPs present in the design and compare it with the report which was provided by the integrator. Hierarchical designs will contain multiple signatures, which can be extracted in form of information trees, thus allowing tracing infringement to the source, as suggested in [1].

Unlike other approaches found in the literature [1, 2, 3, 4], the proposed scheme does not require any modification of the original designs and it is used merely as a verification methodology. Identical signatures are however extremely unlikely to be achieved in significantly different designs. The odds that such an event occurs are

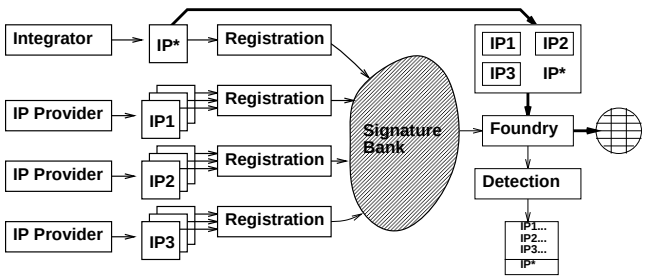


Figure 1: *Registration and detection*

0-7803-5832-X /99/$10.00 ©1999 IEEE.

denominated as P_u. Non-extensive tampering deviates the original signature in predictable patterns, thus allowing for simple error-correction techniques to be applied. Nonetheless, the probability P_m that a signature coincides with one of another design is non-zero. Signatures need be generated so as to keep these two probabilities low, typically less than 10^{-10}.

The paper is organized as follows. The signature generation techniques are outlined in Section 2. Section 3 presents the scheme utilized in the detection phase. Experiments showing the suitability of the approach are discussed in Section 4.

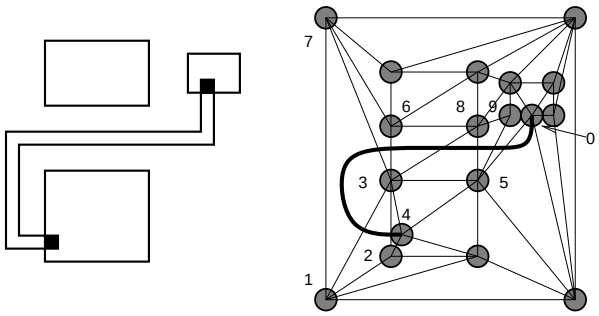


Figure 2: *Bubbles and rough routings*

2 Generating Signatures

Let Σ^* be the set of all strings in a finite alphabet Σ, e.g. $\Sigma = \{0, 1\}$. Assume there exists a compact representation or signature for a given design at some abstraction level. Let $s \in \mathcal{S}$ be one of all possible physical implementations of the design, let σ_s be its signature. Define *signature mapping* $\mathcal{S} \rightarrow \Sigma^* : \mathcal{M}$ as the mapping of a subset of all the layout features onto a signature $\sigma_s = \mathcal{M}(s)$. Let us define $\mathcal{S} \rightarrow \mathcal{S} : \mathcal{F}$ as a mapping which transforms implementation s onto a new implementation $s' = \mathcal{F}(s)$. If $\sigma'_s = \sigma_s$, then $\mathcal{F}(s)$ is said to be *signature-invariant*.

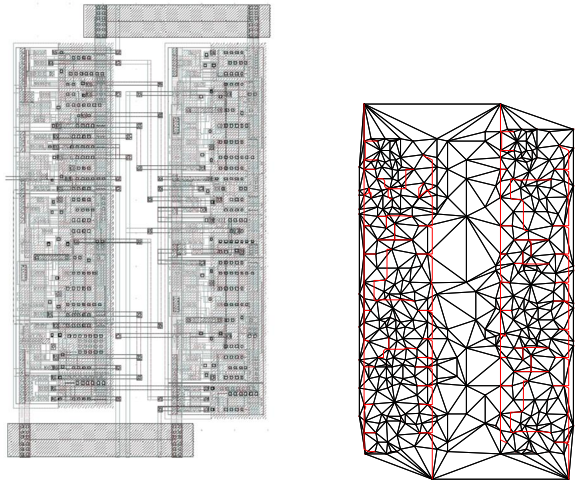
Let us now describe the particular mapping used in this paper to generate design signatures. Let us assume that the granularity of the circuit is given. As a result, the set of fundamental components, such as transistors or universal standard cells, is determined. Call such components *atomic blocks* and Ω their set. In s, every component $\omega \in \Omega$ may have multiple instantiations.

A layout implementation defines a set of all relative positions and orientations of every component instantiation in the circuit. Interconnect can be represented in a similar fashion where components are replaced by pins, Steiner points, and bends. A composition, containing the details of all relative positions and orientations, is called *topology*. Let us now use the layout's atomic blocks, pins, Steiner points, and interconnect bends, which are in turn represented by a set of primitives called *bubbles*, as proposed in [5]. A bubble is a point associated with a given layer. Let B be the set of all bubbles in the design. Every atomic block is mapped onto m distinct bubbles according to a specific mapping $\Omega \rightarrow B : \mathcal{B}$, where m is a finite natural number. For simplicity, but without loss of generality, suppose that m is constant over Ω. Note that $|B|$ grows linearly with the number of atomic blocks and pins.

Paths can be represented by a continuous curve of finite length which begins and ends in a bubble. Such curve is known as *rough routing* [6]. The design rules of a given technology can be seen as minimum spacing constraints between the perimeters of bubbles and paths. Alternatively, after proper scaling of the design rules, one can consider bubbles as points, and paths as curves of zero-thickness. For simplicity we have adopted this convention. Let *topological routing* be an equivalence class of rough routings connecting its pins. Two rough routings of a wire are equivalent when one can be obtained from the other by continuous deformation with no violations of any of the scaled design rules. Assume that every pair of bubbles is connected by an edge, then if a topological routing crosses such an edge, it is said to *intersect topologically* the edge.

If every region in the layout is partitioned in simply connected regions, each containing no bubbles, then such regions are called *simple regions*. Figure 2(a) shows an interconnect and some obstacles, while Figure 2(b) depicts the corresponding partition into simple regions. The rough routing connecting bubble 4 to 0 can be represented in terms of the sequence of all topological intersections. In this case such a sequence is: $\sigma = (\overline{23},\overline{13},\overline{37},\overline{36},\overline{38},\overline{58},\overline{59},\overline{50})$. Note that symbol $\overline{X_i\ X_j}$ represents the topological intersection of the rough routing with the edge spanned by bubbles X_i and X_j. Define E_ℓ as the set of all simple regions in a given layer ℓ and a *planar subset* $T_\ell \subset E_\ell$ as one in which distinct edges do not intersect or they intersect at only one of the vertices. In addition, if T_ℓ has a convex boundary or *convex hull*, it is said to be maximally planar. Under these conditions, T_ℓ is called *triangulation* [7]. Let us now assume that an arbitrary triangulation T_ℓ is in place for each layer. Let B_ℓ be the set of all the bubbles associated with T_ℓ. For convenience, although not needed, let us set four bubbles at the extremities of the union of all the layers, so as to encompass every layer.

Sequence σ is a non-unique representation of all the rough routings associated with the class of this topological routing. Hence, to make such representation resilient to minor modifications, it is necessary to convert it onto a canonical form. This is done simply removing adjacent identical edges, which form so-called *loops*. The unique canonical form of an arbitrary topological routing τ is called *topological signature* σ_τ. The complexity of loop removal is higher when it involves a large number of rough



(a) **(b)**

Figure 3: *(a) Layout; (b) Associated triangulation*

routings. The process in this case must be recursively performed.

Triangulations are not unique. However if the method used to obtain a certain triangulation is an invariant, then the signature is also invariant for a certain design. Figure 3(a) for example shows a simple layout based on standard cells organized in two rows with the corresponding interconnect. Figure 3(b) shows a possible triangulation of the associated topology. The computational scheme and circuit topology determine the final result [7].

The uniqueness of a signature is defined by probability P_u, its robustness by P_m. Topological intersections are unique for a given design and triangulation, while a triangulation is determined by the utilized algorithm and by set B. For each layer the number of possible triangulations grows factorially as $(|B_\ell| - 1)!/3!$, hence it is reasonable to choose a layer ℓ^* which maximizes $|B_\ell|$ over all layers. By a conservative estimate, N_T, the total number of possible triangulations over all layers, is then $N_T \geq (|B_{\ell^*}| - 1)!/3!$. Suppose now that all N_{ℓ^*} topological routings in ℓ^* consist of N_i i-terminal nets, $i = 2, \ldots, N_{max}$. Then, all N_{ℓ^*} topological routings can be represented in terms of N' two-terminal sub-routings, with $N' = \sum_{i=2}^{N_{max}} N_i\,(i-1)$. As a consequence, the number of possible topological signatures can be computed as $N_\sigma \geq N_T \binom{N'}{2}$, hence the estimate of P_u becomes $\overline{P_u} \leq \frac{1}{N_\sigma}$. For example, suppose that for a given design $|B_{\ell^*}| = 20$, $N_{\ell^*} = 10$, $N_2 = 3, N_3 = 5, N_4 = 2$. Then, $N_\sigma \geq (20-1)!171/3! = 3.5\times10^{18}$, hence $\overline{P_u} \leq 2.9\times10^{-19}$.

In the absence of tampering $P_m = 0$, i.e. the signature extracted from a topology matches 100% with the one which is registered in the signature bank. If tampering has occurred, it needs to be modeled in order to properly estimate its effects on P_m. Let us consider the following tampering attempts: (1) routing modification, (2) atomic block modification, and (3) atomic block move and/or addition/deletion. Attempt (1) does not change triangulation, however it may cause changes in the signature. Such changes are of three basic types: symbol addition, deletion and swap. More than one symbol may be involved in the change at any time, however, when this occurs, the change can be modeled in terms of a composition of simple symbol modifications. Attempts (2) and (3) may change the triangulation. However, their effects can be modeled in terms of simple symbol operations.

Define P_r as the probability that a symbol change occurs. Then, the probability that a signature of size t mutates is $P_t = \sum_{j=1}^{t} \binom{|B|}{j} [P_r]^j \times [(1 - P_r)]^{|B|-j}$. Hence, for example, if $t = 1$ and $P_r = 10^{-5}$, then $P_m \leq 9 \times 10^{-6}$.

3 Detecting Signatures

Signature detection consists of the following phases

1. bubble extraction
2. transformation inference
3. bubble matching
4. triangulation
5. signature computation

The initial layout is flattened and all its layers are extracted and deconstructed into polygons or basic standard cells. Using standard slicing techniques [8], the layout is partitioned in rectilinear areas encompassing exactly one atomic block. The complexity of this operation is $O(|\Omega|\ log|\Omega|)$ where $|\Omega|$ is the number of objects in the layout. Using mapping $\mathcal{B}$, the design is entirely converted into a bubble-based representation in $O(|\Omega|)$ time (phase 1).

In order to detect the presence of blocks with known signatures embedded in the design, one has to infer the most likely orientation of every candidate block. This operation is performed by matching complex interconnect patterns present in both the host and the embedded design. Consider the designs of Figure 4. Suppose the interconnects shown in shaded lines are to be used to determine the orientation of the embedded circuit within the host. Let us first catalog all the interconnects present in both layouts in order of size (equal to the number of interconnect segments) in $O(n \log n)$ time. Then, for each pair of interconnects of identical size, a transformation $(\triangle x, \triangle y, \theta, s_x, s_y)$ is derived which maximizes the number of points that can be transformed from the embedded to the host design. Note that s_x, s_y represent a possible scaling operation. Deriving $(\triangle x, \triangle y, \theta, s_x, s_y)$ requires the solution of a system of eight linear equations for each pair of candidate interconnects in the worst case. Then, the most frequently occurring transformation is selected. The solution time of each system of equations is constant,

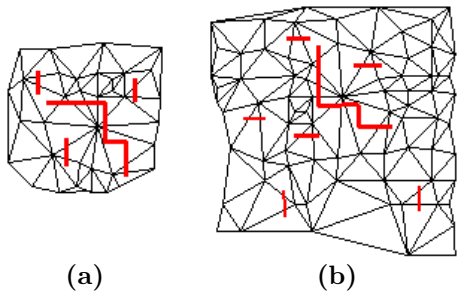


Figure 4: *Transformation inference: (a) embedded, (b) host design*

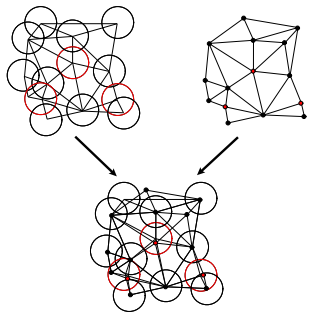


Figure 5: *Principle of range search*

the worst case time complexity is therefore quadratic in the number of interconnects of identical size. (phase 2).

Next, the bubble representation of the host needs to be matched with that of the transformed embedded design. This procedure is accomplished by superimposing the designs and by assigning every bubble in the host to exactly one in the embedded design which minimizes the Euclidean distance. The search is initially performed within a zero range, which is augmented multiple times by a unit length until a neighbor is found. Figure 5 shows the range search process (phase 3).

Finally, using optimal algorithms, a Delauney triangulation is computed in $O(|B|\ log|B|)$ time for both designs [7, p. 241] (phase 4). The line segment intersection algorithm is used for the computation of the edges being intersected by each topological routing. The complexity of this operation is again $O(|B|\ log|B|)$ [7, p. 285]. The signature is derived from this information in a straightforward way (phase 5). In summary, the complexity of entire signature detection process is $O(n\ log\ n)$, where n is the number of atomic blocks, pins and Steiner points in the topology.

4 Results

A complete pass in the flow of Figure 1 was simulated in order to verify the suitability of the approach. The tools utilized in the flow were implemented in C/C++ running

circ.	n_n/ N_n	dev./ IO/nets	ECO density 5 %	ECO density 10 %	re-des.	CPU [s]
s27	2/∞	69/5/96	99.05	96.68	8.24	76.9
s27	3/∞		100	100	7.80	53.0
s27	10/∞		100	100	4.28	43.0
s444	2/∞	709/9/932	100	93.0	10^{-6}	1598
s444	10/∞		100	93.5	10^{-6}	1087
s832	4/∞	1686/37/2127	100	-	10^{-6}	1950
s832	10/∞		100	-	10^{-6}	1620
s1196	10/∞	2105/28/2682	100	96.0	10^{-6}	2383

Table 1: *Signature matching with ECOs and re-design*

under UNIX/LINUX operating systems. All CPU times are referred to a Sun UltraSparc 2 with 256MB of memory. The experiments were based on a set of MCNC 86 and ISCAS 85/89 benchmarks. Each circuit was synthesized and mapped to a SCMOS technology using Sis[9]. Place&route was performed by TimberWolfSC-4.1[10].

To simulate the registration phase, a signature was generated for each benchmark. Then, small modifications were introduced in every benchmark to check whether the signature was resilient to "official" Engineering Change Orders (ECOs) and scaling. Later, a variable number of random non signature-invariant mappings $\mathcal{F}$ were performed on the benchmark's layout so as to maximize the potential damage to the circuit. $\mathcal{F}$ introduced changes on atomic blocks, pins, Steiner points, and nets, uniformly distributed over the entire circuit. Three types of modifications were implemented: (1) translation/rotation, (2) swap, and (3) stretch, aimed at simulating illegal tampering. The signatures associated to the modified designs were compared with the original ones. Finally, the benchmarks were entirely redesigned and the signatures were again compared to the original ones, thus estimating the event that a design could be mistakenly detected even when a "legal" redesign had taken place.

Table 1 reports circuit data, such as device, IO pin, and net count. The signature matching rates are given for several modification densities, simulating an ECO applied to the circuit. The signature was constructed with a minimum net size n_n of 2, 3, 4 or 10 terminals, while no net size upperbound N_n was used. As expected, small ECOs generally resulted in perfect matching, while re-designs resulted in very low matching rates. Moreover, small circuits were less robust to tampering than large ones, due to the lower number of degrees of freedom available to their design.

For the detection phase a large benchmark was selected as the host design. Small benchmarks were embedded, at random locations, in the host. The detection algorithm was run on this example to extract the original signature of the host as well as that of the embedded designs. In various experiments the embedded circuits made up 1% to 10% of the entire host. Finally, tampered circuits

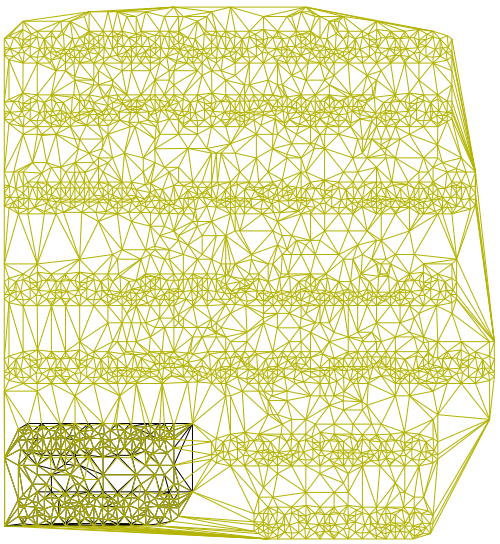


Figure 6: *Detection of embedded circuit*

embedded circ.	host circ.	n_n/N_n	ECO density 0 %	ECO density 10 %	CPU [s]
s27	s1196	$1/\infty$	73.8	73.8	218
s27	s1196	$2/\infty$	72.7	72.7	218
s27	s1196	$5/\infty$	72.7	72.7	218
s1196	-	$1/\infty$	100.0	99.2	1241
s1196	-	$2/\infty$	100.0	99.0	1241
s1196	-	$5/\infty$	100.0	98.6	1241

Table 2: *Signature matching with embedded circuits*

were embedded in the host to verify the robustness of the approach in the presence of multiple levels of tampering. Figure 6 shows an example of a single inclusion of benchmark "s27" into "s444". Table 2 summarizes the results of the detection experiment. Despite the presence of embedded circuits, the host still maintained high signature matching (rows 3-6 in Table 2). The recognition algorithm performed well in identifying both untampered embedded circuits and heavily tampered ones.

5 Conclusions

A method has been proposed for protecting the copyrights of designs in which several virtual blocks, originated from multiple sources, have been integrated. The process consists of two phases. In the first phase, virtual components are mapped onto a signature, which is made public. In the second phase, a signature detection algorithm is applied to a circuit submitted for fabrication to produce a map of all virtual blocks present in the design. The method is effective in detecting and tracing intellectual property infringement *before* fabrication, thus minimizing potential litigation.

6 Acknowledgements

The authors are grateful to Sylvio Triebel for useful discussions and for assistance in the implementation aspects of this work.

References

[1] E. Charbon, "Hierarchical Watermarking in IC Design", in *Proc. IEEE Custom Integrated Circuit Conference*, pp. 295–298, May 1998.

[2] J. Lach, W. H. Mangione-Smith and M. Potkonjak, "FPGA Fingerprinting Techniques for Protecting Intellectual Property", in *Proc. IEEE Custom Integrated Circuit Conference*, pp. 299–302, May 1998.

[3] E. Charbon and I. Torunoglu, "Watermarking Layout Topologies", in *Proc. IEEE Asia South-Pacific Design Automation Conference*, January 1999.

[4] J. Lach, W. H. Mangione-Smith and M. Potkonjak, "Robust FPGA Intellectual PropertyProtection through Multiple Small Watermarks", in *Proc. IEEE/ACM Design Automation Conference*, pp. 831–836, June 1999.

[5] T. Whitney, *Hierarchical Composition of VLSI Circuits*, PhD thesis, California Institute of Technology, 1985.

[6] J. Valainis, S. Kaptanoglu, E. Liu and R. Suaya, "Two-Dimensional IC Layout Compaction Based on Topological Design Rule Checking", *IEEE Trans. on Computer Aided Design*, vol. CAD-9, n. 3, pp. 260–275, March 1990.

[7] F. P. Preparata and M. I. Shamos, *Computational Geometry. An Introduction*, Springer, second Edition, 1988.

[8] R. H. J. M. Otten, "Automatic Floorplan Design", in *Proc. IEEE/ACM Design Automation Conference*, pp. 261–267, June 1982.

[9] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton and A. L. Sangiovanni-Vincentelli, "SIS: A System for Sequential Circuit Synthesis", Memorandum UCB/ERL M92/41, UCB, Univ. of California, Berkeley, CA 94720, May 1992.

[10] C. Sechen and A. L. Sangiovanni-Vincentelli, "Timberwolf3.2: A New Standard Cell Placement and Global Routing Package", in *Proc. IEEE/ACM Design Automation Conference*, pp. 432–439, 1986.