LOSSLESS COMPRESSION ALGORITHMS FOR POST-OPC IC LAYOUT

Allan Gu and Avideh Zakhor

Department of Electrical Engineering and Computer Sciences
University of California at Berkeley, CA 94720, USA

ABSTRACT

An important step in today's Integrated Circuit (IC) manufacturing is optical proximity correction (OPC). While OPC increases the fidelity of pattern transfer to the wafer, it also results in significant increase in IC layout file size. In this paper, we develop two techniques for compressing post-OPC layout data while remaining compliant with existing industry standard data formats such as OASIS and GDSII. The motivation for doing so is for the resulting compressed files to be viewed and edited by any industry standard CAD tools without a decoder. Our approach is to eliminate redundancies in the representation of the geometric data by finding repeating groups of polygons between multiple cells as well as within a cell. We refer to the former as "inter-cell sub-cell detection" and the later as "intra-cell sub-cell detection". Both problems are NP hard, and as such, we propose two sets of greedy algorithms to solve them. We show the results of our proposed inter-cell and intra-cell algorithms on actual 90nm, 130nm, and 180nm IC layouts.

Index Terms— IC layout, compression, OPC, repeating geometries

1. INTRODUCTION

As the semiconductor industry moves toward denser designs with smaller feature sizes, pattern transfer from reticles to wafers, referred to as lithography, becomes more challenging. To correctly fabricate these circuits using current lithographic machines, Resolution Enhancement Techniques (RET) such as optical proximity correction (OPC) are routinely performed on the layout. Denser circuit design plus increased usage of RET have resulted in significant explosion of layout data volume. Specifically, The International Technology Roadmap for Semiconductors shows that the size of a single layer of an uncompressed fractured layout is likely to exceed 400 Gigabytes in 2007 [1]. In particular, OPC is a major contributor to the expansion of layout data volume. OPC destroys hierarchical structures in layouts, and adds vertices to polygons causing over $10\times$ increase in file size.

Layout data are commonly encoded using industry standard GDSII or OASIS binary file format. Both formats use the BackusNaur Form metasyntax to express data records, which describes the geometries created by IC designers. For example, a record describing a polygon contains a list of vertices and a 2D coordinate indicating the location of the polygon with respect to some coordinate system.

There exist compression algorithms to reduce the mask data size in the rasterized domain for direct write lithography systems [2, 3]. There are also algorithms which can be adapted to compress hierarchical IC layout data. Specifically, Chen *et al.* [4] have investigated algorithms to compress dummy fills in IC layouts which exhibit high degree of spatial regularity. Veltman and Ashida have proposed a compression technique for E-Beam writers by finding a set of polygons with identical repetitions [5].

In this paper, we propose two compression techniques to reduce the layout data size. Our techniques guarantee that the resulting compressed layouts remain compliant with GDSII and OASIS, and can be read by any CAD tool without a decoder. Our approach is to eliminate redundancies in the representation of the geometric data by finding repeating groups of polygon between multiple cells as well as within a cell. We refer to the former as "inter-cell sub-cell detection (InterSCD)" and latter as "intra-cell sub-cell detection (IntraSCD)". Both problems are NP hard, and as such, we propose two sets of greedy algorithms to solve them. Section 2 describes the problem of finding repeating groups of geometries within a cell and between multiple cells. In Section 3, we present our algorithms to solve these two problems. Section 4 discusses experimental results on industrial post-OPC layout data, and compares the performance of the proposed algorithms with GZIP.

2. SUB-CELL DETECTION PROBLEM FORMULATION

We begin by defining few terminologies that are used throughout the paper. We define rectangle, trapezoid, polygon, and placement as geometries. A placement is a reference to a cell in the layout, and associated with a placement is a transformation. A cell is a collection of geometries, and a sub-cell is a subset of the geometries in a cell. Two geometries are equal if they have the same list of vertices; for placements, they need to reference the same cell and have the same transformation. The compression ratio (CR) is the ratio of the size of the post-OPC OASIS file to the size of its compressed version.

[1]This work was supported jointly by SRC contract 2005-OC-460 and DARPA contract W911NF-04-1-0304.

2.1. Intercell sub-cell detection

In OASIS, geometries are defined each time they occur in a cell. If a group of three geometries is in N different cells, then there are $3N$ definitions of these geometries when only 3 definitions would suffice. By detecting this repeating group of three geometries, it is possible to create a cell from them which can then be referenced by each of the N cells with a placement operator.

Intercell Sub-cell Detection Problem: Given m cells, $\{C_1, C_2, ..., C_m\}$, find the sub-cell, (SC_r), which maximizes $|SC_r| * r$, for $2 \leq r \leq m$.

A sub-cell SC is said to occur in a cell C if there exists a transformation L that maps every geometry in SC to another geometry in C. $|SC_r|$ denotes the number of geometries in the sub-cell, and r is the number of cells that SC_r occurs in. This problem is NP hard since it is a special case of the largest common point set (LCP) problem [6].

2.2. Intracell sub-cell detection

Representing N instances of a geometry in OASIS requires one geometry definition and N two dimensional coordinates. Compression is achieved by finding a sub-cell which occurs multiple times within the cell. For instance, 4 polygons occuring N times in a cell would require 4 definitions and $4N$ coordinates to represent. Grouping the 4 polygons together into a cell would only require N coordinates rather than $4N$ coordinates.

Intracell Sub-cell Detection Problem: Given a cell, C, find the sub-cell SC_r which maximizes $|SC_r| * r$, for $2 \leq r \leq m$, and the maximum Euclidean distance between any two geometries in SC_r is less than equal to $dist$.

We restrict the maximum Euclidean distance between two geometries because most circuit designs are created by connecting smaller functional circuits together, and the smaller circuits are limited in size. A sub-cell SC occurring in r locations implies that there exist r transformations, $T_1, T_2, ..., T_r$ such that $T_i(SC)$ maps uniquely to a group of geometries in C. m is the maximum number of geometries that are repeated in C. This problem can be shown to be NP hard by a reduction from the 1-D LCP problem [7].

3. SUB-CELL DETECTION ALGORITHMS

InterSCD and IntraSCD are both NP hard problems, and cannot be solved optimally within a reasonable time. In this section, we describe two greedy algorithms to solve them. The approach described in this paper only detects groups of geometries that are translational invariant, but not rotational invariant. We describe an extension to the IntraSCD algorithm that addresses rotational invariance in [7]. Future research will address rotational invariance of the InterSCD algorithm.

3.1. Intercell sub-cell detection algorithm

Before detecting a common sub-cell among a large collection of cells, we begin by grouping cells that may have a common group of geometries using hierarchical clustering [7]. This way, cells that do not share any geometries with other cells are eliminated from further consideration.

Having obtained a collection of clusters through hierarchical clustering, we now find the sub-cell in each cluster which maximizes $|SC_r| * r$, where r is the number of cells the sub-cell occurs in for that cluster, and $|SC_r|$ is the number of geometries that the sub-cell SC_r contains. We start by choosing two cells, C_i and C_j, that are closest in terms of the distance metric described in [7]. Exhaustive search is used to find the largest sub-cell that is common to both cells under translation [7]. The largest common group of geometries is taken as the initial sub-cell if the number of geometries exceeds some threshold. Otherwise, another pair of cells whose distance is the next closest are choosen. Once an initial sub-cell, SC, is selected, we determine the distance between SC and the rest of the cells in the cluster according to the distance metric described in [7]. The cell that is the closest to SC is choosen, and exhaustive search is applied again to find the largest sub-cell SC_i that is common to both SC and the i^{th} cell. Specifically, for the i^{th} cell we test to see whether $|SC_i| * i > |SC| * (i - 1)$, and $|SC_i| \geq threshold$, in order to set $SC \leftarrow SC_i$. Otherwise, the i^{th} cell is removed from the cluster and further consideration. This continues until all of the cells within the cluster have been visited.

Figure 1 shows an example of how the above approach works. After the hierarchical clustering step, cells A, B, C, and D are assumed to be grouped together in a cluster. Cells A and B are the closest with 6 geometries in common. We then apply the exhaustive search to find the largest group of geometries that occurs in cells A and B, and set it as the sub-cell SC shown in Figure 1(b). Cell C and SC are the closest, and SC_2 shown in Figure 1(c) is the sub-cell with the most number of shared geometries between cell C and SC. Finally SC_3 shown in Figure 1(d) is the sub-cell with the most number of shared geometries between SC_2 and cell D. However, SC_3 is not used because $|SC_3| * 4 = 12$ which is not greater than $|SC_2| * 3 = 12$.

3.2. Intracell sub-cell detection algorithm

We have developed a greedy algorithm that grows the sub-cell at each iteration to solve the IntraSCD problem. The basic idea behind the algorithm is to select an initial polygon as an initial cell, and to add more polygons to the cell until there is no additional benefit in adding more polygons. At each step of the iteration, we choose the group of geometries such that $|SC_i| * numInst_i$ is maximized, where $|SC_i|$ is the number of geometries in the sub-cell at the i^{th} iteration, and $numInst_i$ is the number of instances of SC_i in the cell. The algorithm stops adding more polygons when $|SC_i| * numInst_i \leq |SC_{i-1}| * numInst_{i-1}$.

The algorithm starts by ranking all the geometries according to the number of instances of the geometry in the cell. Then the geometry, G_{max}, with the most number of instances is selected. For each instance of G_{max}, all possible groups

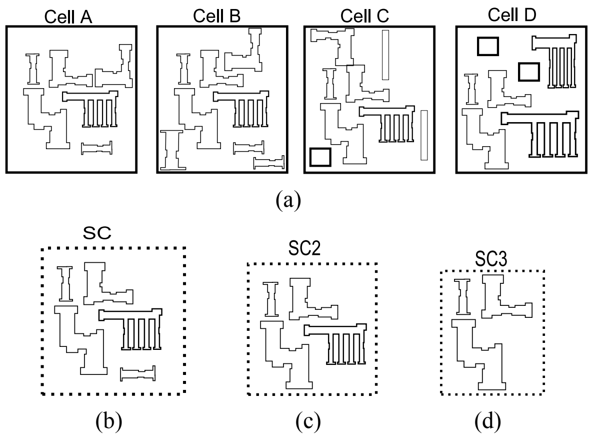

Fig. 1. Intercell sub-cell detection example. (a) Cell cluster; (b) sub-cell of cell A, B; (c) sub-cell of SC, cell C; (d) sub-cell of SC2 and cell D

of 2 or 3 geometries are created using G_{max} and its neighbors [7]. We select the most frequently occuring group of 2 or 3 geometries, SC, provided the number of instances of that group is greater than some threshold. Additional geometries are added to the group in order to determine whether it is worthwhile to expand its size. We consider SC as a single entity, and apply the same iteration step described above. The algorithm stops adding more polygons on the i^{th} iteration if $|SC_i| * numInst_i \leq |SC_{i-1}| * numInst_{i-1}$. Once the iteration has ended, a new cell containing the geometries of SC_{i-1} is created and placements at all the locations in the cell that SC_{i-1} occurs at are created. The above process is repeated until all of the geometries have been visited.

Figure 2 shows an example of running the IntraSCD algorithm on a cell with 31 different geometries. Initially in Figure 2(a), we select the polygon with 5 instances called SC_0 and examine all its possible combinations of 2 and 3 geometries. Figure 2(b) shows the group of three polygons that have the most benefit among all the combinations after the 1^{st} iteration. Since $|SC_0| * numInst_0 < |SC_1| * numInst_1$, we continue the iteration. At the end of the 2^{nd} iteration, another polygon is added to SC_1 resulting in a group of 4 polygons as shown in the top sub-cell in Figure 2(c) which we call SC_2. Figure 2(c) also shows two other groups of geometries considered in the second iteration. However, these groups only occur once in the cell and are not selected. SC_2 with 4 geometries appearing on the top of Figure 2(c) is selected because it is the one that maximizes $|SC| * numInst$. The iteration continues since $(|SC_2| * numInst_2 = 16) > (|SC_1| * numInst_1 = 12)$. On the third iteration, the algorithm attempts to add more geometries to SC_2. However $(|SC_3| * numInst_3 = 7) < 16$, and therefore the process stops. Figure 2(d) shows the result at the end of the iterations, where the resulting sub-cell has replaced the repeating geometries in the cell.

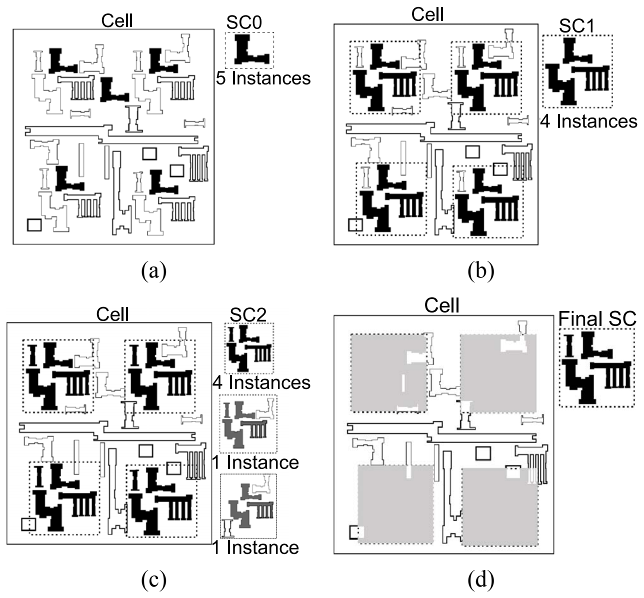

Fig. 2. Intracell sub-cell detection example. (a) 0^{th} iteration; (b) 1^{st} iteration; (c) 2^{nd} iteration; (d) result

Table 1. InterSCD compression ratio. File sizes are in bytes.

	Post-OPC Size	InterSCD Size	CR
Poly (L1)	6,391,097	2,793,277	2.288
Active (L1)	3,496,377	1,777,757	1.967

4. RESULTS

We apply the above InterSCD and IntraSCD algorithms on actual industrial post-OPC layouts. The first data set consists of the Poly and Active layers for a 3.5mm $\times$ 3.5mm chip with 180nm feature size. The OPC is done by the layout owner with industry standard OPC software. The second data set consists of the Poly, Metal 1, and Metal 2 layers from a 8mm $\times$ 8mm and 4.3mm $\times$ 4.3mm chips with 130nm feature size. The third data set consists of the Poly, and Active layers from a 1.4mm $\times$ 1.4mm and 1.8mm $\times$ 1.8mm chips with 90nm feature size. We performed OPC correction on the second and third data sets with standard recipes using a different OPC software from another major vendor.

We have found that for the 1^{st} data set the InterSCD works well, while applying IntraSCD does not result in noticeable gain. We notice that many of the post-OPC cells from the first layout data set are much smaller than those from the second and third data sets. Therefore, IntraSCD, which detects similar groups of polygons within a cell, does not result in much gain on the first layout data set with small cells. Table 1 shows the InterSCD compressed file sizes in bytes encoded in OASIS format for the first data set. As shown, the average compression ratio is around 2X for the two layers.

Similarly, IntraSCD works well on the 2^{nd} and 3^{rd} data sets, while InterSCD results in little gain. Table 2 shows the results of applying IntraSCD on the second and third layout data sets. The compression ratio ranges from 1.80 to 2.46

Table 2. IntraSCD compression ratio. File sizes are in bytes.

	Post-OPC Size	IntraSCD Size	CR
Poly (L2a)	2,413,460	977,294	2.469
Poly (L2b)	1,036,664	576,491	1.798
Poly (L3a)	9,189,288	4,905,897	1.873
Poly (L3b)	34,515,762	18,960,928	1.820
Metal 1 (L2a)	2,490,423	1,791,495	1.390
Metal 1 (L2b)	1,194,192	1,060,746	1.126
Metal 2 (L2a)	1,444,367	1,143,360	1.263
Metal 2 (L2b)	947,981	775,561	1.222
Active (L3a)	9,666,584	6,899,025	1.401
Active (L3b)	35,945,586	23,209,262	1.549

Table 3. Comparing the CR of GZIP to InterSCD.

	GZIP CR	InterSCD CR	InterSCD+GZIP CR
Poly (L1)	7.789	2.288	8.987
Active (L1)	8.603	1.967	9.150

for the Poly layers, and 1.40 to 1.55 for the Active layers. However, the compression ratio for the Metal layers is rather low and in the range of 1.12 to 1.39. The Metal layers contain many polygons with only a few instances. Naturally, if a polygon, P, has few instances, then there are only few instances of a group of polygons containing P.

GZIP [8], a popular lossless compression software, is commonly used to compress GDSII and OASIS layout files. Table 3 compares the compression ratios of GZIP and InterSCD. As seen in the 2^{nd} column of Table 3, GZIP performs well, achieving a compression ratio of 8.6 for Active (L1). The compression ratio of InterSCD shown in the 3^{rd} column of Table 3 is lower than that of GZIP. This is expected since InterSCD compressed files are OASIS format compliant, and as such, do not employ any entropy coding techniques. However, as shown in the 4^{th} column of Table 3, InterSCD files can be further compressed by applying GZIP to them for situations in which OASIS compliancy is not important. As seen in the 4^{th} column of Table 3, the compression ratio of InterSCD follow by GZIP is higher than GZIP by itself.

GZIP also performs better than IntraSCD for layouts with larger cells. However, as seen in the 2^{nd} and 3^{rd} columns of Table 4, the compression ratio of IntraSCD is much closer to the compression ratio of GZIP for larger post-OPC layout file sizes such as Active(L3b) and Poly(L3a). Specfically, for the largest layouts i.e. Layout 3b, the compression ratios are 1.55 and 1.77 for the Active layer, and 1.82 and 2.34 for the Poly layer for IntraSCD and GZIP respectively. Similar to InterSCD, IntraSCD compressesed files can be further processed by GZIP to achieve better compression at expense of not being OASIS compliant. As expected, Column 4 of Table 4 shows the compression ratio of IntraSCD followed by GZIP is higher than GZIP for all of the layouts, with the improvements ranging from 42% to 76% for Layouts 3a and 3b.

Table 4. Comparing the CR of GZIP to IntraSCD.

	GZIP CR	IntraSCD CR	IntraSCD+GZIP CR
Poly (L2a)	4.972	2.470	6.187
Poly (L2b)	3.875	1.798	4.267
Poly (L3a)	2.531	1.873	4.447
Poly (L3b)	2.366	1.820	4.179
Metal 1 (L2a)	3.003	1.390	3.786
Metal 1 (L2b)	2.764	1.126	2.898
Metal 2 (L2a)	2.96	1.263	3.110
Metal 2 (L2b)	2.911	1.222	3.043
Active (L3a)	1.875	1.401	2.677
Active (L3b)	1.768	1.549	2.866

5. CONCLUSION AND FUTURE WORK

We have presented two lossless compression algorithms called InterSCD and IntraSCD for post-OPC IC layout data. In addition to being lossless, the resulting compressed files are fully compliant with industry format i.e. OASIS, which means they can be viewed by any CAD editing tool without a decoder. The algorithms find redundancies in terms of repeating geometries within a cell and between cells. We have extended the IntraSCD algorithm in [7] to include rotational and reflectional invariance resulting in 5 to 6 percent increase in compression ratio for Layouts 3a and 3b. In addition, IntraSCD algorithm can also be used to rediscover hierarchy in flattened layout [7]. Future work involves applying the algorithms on larger layouts, and extending InterSCD to handle rotations and reflections.

6. REFERENCES

[1] "ITRS 2005 Edition, Lithography," Available at http://www.itrs.net/Common/2005ITRS/Litho2005.pdf.

[2] V. Dai and A. Zakhor, "Lossless compression of VLSI layout image data," *IEEE Trans. on Image Processing*, vol. 15, no. 9, pp. 2522–2530, 2006.

[3] H. Liu, V. Dai, A. Zakhor, and B. Nikolic, "Reduced complexity compression algorithms for direct-write maskless lithography systems," in *Proceedings of SPIE, Vol. 6151, Emerging Lithographic Technologies X*, 2006, pp. 632–645.

[4] Y. Chen, A. Kahng, G. Robins, A. Zelikovsky, and Y. Zheng, "Compressible area fill synthesis.," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 24, pp. 1169–1187, 2005.

[5] R. Veltman and I. Ashida, "Geometrical library recognition for mask data compression," in *Proceedings of SPIE - Vol. 2793, Photomask and X-Ray Mask Technology III*, 1996, pp. 418–426.

[6] T. Akutsu and M. M. Halldorsson, "On the approximation of largest common subtrees and largest common point sets," in *Proceedings of the 5th International Symposium on Algorithms and Computation*, 1994.

[7] A. Gu and A. Zakhor, "Lossless compression algorithm for hierarchical IC layout data," in *Proceedings of SPIE - Vol. 6520, Optical Lithography XX*, 2007, pp. 652017–1 – 652017–17, http://www-video.eecs.berkeley.edu/papers/agu/spie_2007.pdf.

[8] P. Deutsch, "Gzip file format specification," RFC 1952, May 1996.