Layout Compaction for Yield Optimization via Critical Area Minimization

Youcef Bourai
Electrical Engineering Department
University of Washington Box 352500
Seattle, Washington 98195, USA.
e-mail:ybourai@ee.washington.edu

C.-J. Richard Shi
Electrical Engineering Department
University of Washington Box 352500
Seattle, Washington 98195, USA.
e-mail:shi@ee.washington.edu

ABSTRACT

This paper presents a new compaction algorithm to improve the yield of IC layout. The yield is improved by reducing the area where the faults are more likely to happen known as critical area. Instead of assuming that the critical area could probably be present everywhere in the layout, the algorithm first finds where this area can actually exist, and then attempts to minimize it. The algorithm takes benefit from a fast multi-layer critical area computation to extract the rectangles that compose it. Afterwards, the extracted rectangles are involved into the layer minimization process which is the second phase of the compaction procedure to minimize their area. A new formulation of the layer minimization problem is used in such a way that the critical area minimization adds neither extra variables nor extra constraints to the original compaction algorithm. The algorithm has been tested on actual layouts.

1. INTRODUCTION

Throughout the manufacturing process of VLSI circuits many point defects can be introduced into the IC layers during the lithography stage. These point defects may be missing patterns that cause opens in the circuits, or extra patterns that can cause short circuits.

Nowadays, advances in manufacturing process with mature CAD tools allow to integrate over million transistors on a single chip. At the same time, the risk of introducing more point defects has increased due in one hand to the augmented number of lithography levels and in another hand to the augmented density of the circuits. Thus, the challenge of ever-increasing complexity of IC can be met only if the yield on the chip is maintained at a profitable level, otherwise the VLSI systems will not be longer commercially viable. To achieve a higher yield for the highly integrated circuits, modern CAD tools must consider the yield during design steps, during which not only the compaction tool must look for area optimization but also must consider how various elements are arranged within the given area. Therefore, modern compactors need to be provided with yield optimization. Many yield models have been proposed by the literature. But it has been found that the model given by (1) matches empirical results better than other models do.

$$Y = Y_0 / \left(1 + dA\theta/\alpha\right)^{-\alpha} \qquad (1)$$

where Y is the yield of the chip, Y_0 is the gross yield factors, d is the average number of defects per unit area, A is the area of the die, θ is the probability that a defect will result in a circuit fault, and α is the clustering parameter. The product $A\theta$ is defined as the area that is more sensitive to defects than the other areas or the critical area. In other words, the critical area for the defects of certain size s is defined as the area in which the center of a defect must fall in order to cause a circuit failure. Therefore, by reducing the size of the critical area, the sensitivity of the layout to point defects will be reduced so the yield will be enhanced. Only recently have researchers given attention to point defects while developing tools for compaction. The first significant work has been reported by [1], where a set of local rules have been proposed for contacts, metal and polysilicon layers for yield enhancement. However, the techniques involved are not general enough to be applied on the regular physical layout compaction. In the second method reported by [3], a heuristic algorithm increases the spacing of layout by changing the positions of only the objects off the critical path, while layout area is maintained at its minimum area. This heuristic however does not guarantee optimum yield. The method that considers tightly the fault probability is described in [2]. This method is a variant of the graph based compaction algorithm, where the cost of each edge connecting two vertices is the fault probability due to the spot defect between the two objects represented by the two vertices. The cost is described by a function of the distance between the two objects. The function in turn is approximated by a piece-wise linear approximation. The vertex positions of the modified graph, which minimize the sum of the new edge costs, are found using the enhanced network flow algorithm. In this method the critical area is not addressed directly. Instead, an arc is added by default between each two vertices whether a critical area does exist in between or does not. This general suspicions of the critical area existence between each two objects make the compaction algorithm more complex. In this paper a new algorithm for the optimization of yield during the compaction stage is presented. Minimizing the critical area, which is directly addressed by computing its amount and then reducing it while reducing the total layer area, optimizes the yield. The paper is presented as follows: in section 2 a method to extract the critical area of VLSI circuits is presented. In section 3 the algorithm is discussed in details. In

section 4 some results are discussed. Finally, section 5 concludes the paper.

2. MULTI-LAYER CRITICAL AREA COMPUTATION

As defined above the critical area is the area where the center of a defect must fall to cause a fault. Its correct estimation plays an important role in layout sensitivity to spot defect and yield prediction. In most cases, two adjacent patterns in the same mask layers will cause a critical area. However, in the case that two patterns are electrically equivalent, i.e they are connected elsewhere by other patterns, the critical area between the two patterns is not realistically possible [8]. In order not to count these "false critical areas" as if it would be calculated in the single layer analysis, the multi-layer analysis strategy for an accurate critical area computation is presented. The corner-stitching data structure [6] is involved into the critical area computation, because the data structure is especially powerful in finding the neighboring information of a geometrical patterns which will possibly intersect with the pattern by a spot defect. A potential critical area between each pair of patterns can be computed as follows: For each pattern P_i in a layer L, let (x_{lb},y_{lb}) be the left-bottom coordinate of the pattern and (x_{tr},y_{rt}) be the top-right coordinate. Assuming that the defect size is D_s, an area enumeration procedure is performed to find all patterns (or solid tiles) located in the area $[(x_{lb}-D_s,y_{lb}-D_s),(x_{tr}+D_s,y_{tr}+D_s)]$. If a pattern is tagged with the same net number as P_i's, the pattern is skipped. Suppose that the patterns found are S_i, $1 \le i \le k$, the critical area $C(P_i, S_i)$ between S_i and P_i can be quickly obtained by expanding their boundaries by $D_s/2$, as illustrated by figure 1. Afterwards, this critical area is inserted into an extra corner-stitching plane where only the critical rectangles are represented as solid tiles. The procedure is repeated over all unvisited patterns. Two points make the computed critical area accurate [8].

1. A pattern pair with the same net number is skipped to avoid computing the "false critical areas"
2. By definition the corner stitching data structure maintains non-overlapping tiles. Thus during the insertion of a new critical area tile into its plane, if the tile overlaps the already existing tiles, the overlap area is cut off from the critical area plane. Therefore, the duplication of the critical area is prevented.

3. COMPACTION ALGORITHM WITH YIELD OPTIMIZATION

3.1 Basic Compaction Algorithm (BCA)

It has been proved that the compaction problem is of NP-Hard complexity [7]. In our strategy, the problem is relaxed to become two iterations of a one-dimensional process. It successively compresses the layout in the horizontal dimension and then in the vertical dimension. As the procedure used in each dimension is the same, we will focus our description in the horizontal direction. The compaction algorithm is designed as a two-stage algorithm. In the first stage, the algorithm seeks to minimize the area of the whole layout. This problem can be formulated by the LP (Linear Programming) expression:

$$\min \sum_{i=1}^{N_e} (x_i - x_o) \qquad (2)$$

Subject to a set of constraints C

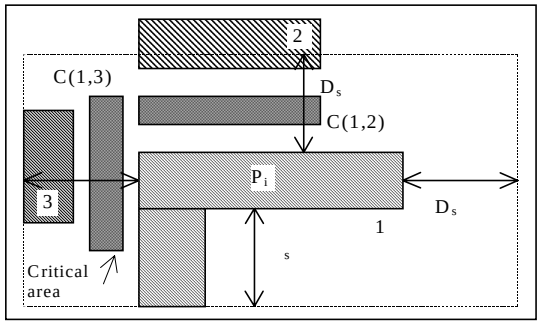

Figure 1: Critical area computation.

where x_i is the abscissa of a vertical edge of a rectangle and x_0 is the abscissa of the most left bound, C is the set of constraints describing the design rules, and N_e is the number of edges. Although the solution to (2) yields the minimum area of the whole chip, it does not guarantee that the areas of the layers are minimized as well. This is due to the fact that this stage can be seen as of placing a piece of strong magnet that attracts edges to the left of the layout as far as the design rules allow. But not all the edges are constrained. The unconstrained edges induce a set of non-constraining rectangles which will have an extra slack room where to expand. This extra degree of freedom leads to unnecessary long rectangles which could therefore leads to an electrically poor layout and to a high defect sensitive layout too. Fortunately, the problem can be tackled in stage II, where the algorithm seeks to minimize the area of layer rectangles. This problem can be formulated by the following LP expression:

$$\min \sum_{i=1}^{N_R} \alpha^i \left(x^i{}_r - x^i{}_l\right)\left(y^i{}_t - y^i{}_b\right) \qquad (3)$$

Subject to the same set of constraints C and the new constraint : $X_R - X_L = W$

Where X_R (res. X_L) is the right (res. left) bound of the layout, W is the solution of (2) and N_R is the number of rectangles. Note that the cost function to be minimized in expression (3) is a weighted sum of areas. By assigning different values to the weighting coefficients α^i, the algorithm can control the minimizing priority among the rectangles as illustrated by figure.2.

The BCA algorithm is depicted by figure.3.

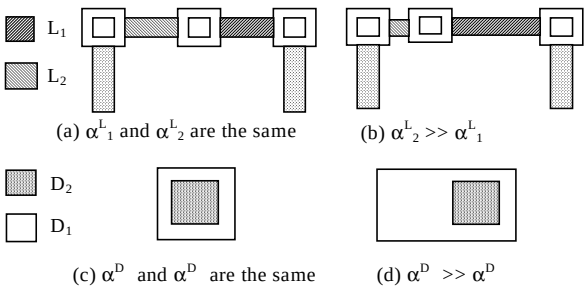

Figure 2: Weighted resizing.

BCA()
{
C = Generate_Constraints();

$$resolve\begin{cases} W = \min \sum_{i=1}^{N_e} (x_i - x_o) \\ \text{subject to } C \end{cases}$$

$$resolve\begin{cases} \min \sum_{i=1}^{N} \alpha^i \left(x^i{}_r - x^i{}_l \right) \\ \text{subject to } C \text{ and } X_R - X_L = W \end{cases}$$

}

Figure.3: Basic Compaction Algorithm.

3.2 Modified Compaction Algorithm (MCA)

During its second stage the BCA seeks to minimize the rectangles of each layer. Since the critical area is a set of rectangles in a separate plan, they are subject to be minimized too. Therefore, the expression (3) becomes:

$$\min \sum_{i=1}^{N_R} \alpha^i \left(x^i{}_r - x^i{}_l \right)\left(y^i{}_t - y^i{}_b \right) + \min \sum_{i=1}^{N_C} \alpha^{crt} \left(x^{crt}{}_r - x^{crt}{}_l \right)\left(y^{crt}{}_t - y^{crt}{}_b \right) \qquad (4)$$

Subject to C and $X_R - X_L = W$.

where N_c is the number of critical rectangles. Since during the horizontal pass, the height of the rectangles do not play any role, they behave like constants. Therefore, the expression (4) can be rewritten as follows:

$$\min \sum_{i=1}^{N_R} \beta^i \left(x^i{}_r - x^i{}_l \right) + \min \sum_{i=1}^{N_C} \beta^{crt} \left(x^{crt}{}_r - x^{crt}{}_l \right) \qquad (5)$$

subject to C and $X_R - X_L = W$.

By construction of the critical area *C(u,v)* between pattern *u* and pattern *v* as mentioned in §2 and by figure.4, the abscissa of its right (res. left) edge $x^{crt}{}_r$ (res. $x^{crt}{}_l$) can be written in function of the right abscissa of pattern *u* (res. left abscissa of pattern *v*) as follows:

$$\begin{cases} x^{crt}{}_r = x^u{}_r + D_s/2 \\ x^{crt}{}_l = x^v{}_l - D_s/2 \end{cases} \qquad (6)$$

Thus: $x^{crt}{}_r - x^{crt}{}_l = x^u{}_r - x^v{}_l + D_s$ (7)

Pattern *u* and pattern *v* are easily found by the area enumeration procedure in the range $[(x^{crt}{}_l - D_s/2, y^{crt}{}_l - s/2), (x^{crt}{}_r + D_s/2, y^{crt}{}_r + D_s/2)]$, see [6]. Finally, the expression (5) can be written:

$$\min \sum_{\substack{i=1 \\ i \neq u \\ i \neq v}}^{N_R} \beta^i \left(x^i{}_r - x^i{}_l \right) + \min \sum_{i=1}^{N_C} \beta^{crt} \left(x^u{}_r - x^v{}_l \right) \qquad (8)$$

subject to C and $X_R - X_L = W$.

To minimize the second sum of the expression (8) $x^u{}_r$ must take its minimum value while $x^v{}_l$ must take its maximum value. This can be thought as of shifting $x^v{}_l$ to the right and $x^u{}_r$ to the left in such a way that not only minimizing the critical area but also minimizing the rectangles *u* and *v* as well, as depicted by figure.4. Such minimization is still the aim of the second phase of the BCA. By assigning a big value to β^{crt} this sum will be minimized first. By writing the coordinate of the critical as a function of the coordinate of the actual layer the MCA adds neither extra variables nor extra constraints keeping the complexity of the BCA unchanged. The MCA algorithm is depicted by figure.5.

4. RESULTS

The compaction method has been tested on the benchmarks of Table.1. To compare the results of the original compactor and the modified compactor, the benchmarks are laid out twice. First using the original compactor and second using the modified compactor. The EDAM system [9] is used to obtain the data concerning the failure probability of both layouts. In this paper, we are interested in computing the layout sensitivity instead of the failure probability, because it is believed that a low layout sensitivity implies small failure probability. A defect size of 4 µm has been used to evaluate the critical areas of the benchmark layouts. Therefore, the defect size is large enough to reflect meaningful layout sensitivities. The effect of the new algorithm is shown in the last column of the table.1. According to the data, we have found out that the layout sensitivity can be decreased if the yield optimization is taken into account during the compaction.

5. CONCLUSION

A compaction algorithm for yield enhancement has been presented. Unlike its predecessors that consider by default that the area where the spot defects are susceptible is spread over the whole layout and tempt to minimize it, our algorithm computes the actual critical area and tempts to reduce it. By targeting the actual critical area the algorithm drastically reduces the run-time. Combining the multi-layer method to compute the critical area with a suitable formulation of the second phase of the compaction algorithm has allowed to upgrade the original compaction algorithm in order to take into account the yield enhancement without neither including extra variables nor extra constraints, keeping the complexity of the original compaction algorithm unchanged.

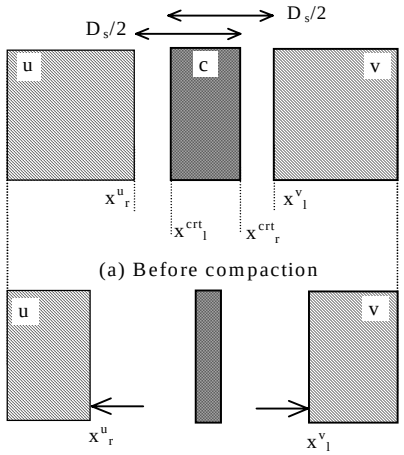

Figure 4 Critical area minimization via layer minimization

MCA()

{

C = Generate_Constraints();

$$resolve\begin{cases}\min \sum_{i=1}^{N} \alpha^i \left(x^i{}_r - x^i{}_l\right) \\ \text{subject to } C \text{ and} \\ X_R - X_L = W\end{cases}$$

$S_c = \varnothing$;

/ Generate the critical area S_c */*

critical_area$_i$ = find_critical_area();

$S_c = S_c \cup critical\_area_i$;

/ for each critical_area$_i \in S_c$ enumerate all the tiles that are responsible for this critical area. Their edges are stored in the list of critical edges E*/*

for (each critical_area$_i \in S_c$)

{

edge = find_critical_edge(critical_area$_i$);

E = E$\cup$ edge;

|E| = |E|+1;

}

$$resolve\begin{cases}\min \sum_{i=1}^{N_R} \beta^i \left(x^i{}_r - x^i{}_l\right) + \min \sum_{j=1}^{|E|} \beta^{crt} \left(x^u{}_r - x^v{}_l\right) \\ \text{subject to } C \text{ and } X_R - X_L = W\end{cases}$$

}

Figure.5: MCA algorithm.

Name	Size	Run-time increase	Sensitivity reduction
Sram	1K	1.67%	12%
Inv6	2K	5.04%	9.7%
C17	10K	0.10%	8.97%
Nor30_a	18K	0.17%	3.26%
Mux21	29K	0.25%	5.33%
Pixel4x4	35K	0.55%	5.28%

Table.1: Layout sensitivity reduction on spot defect.

6. ACKNOWLEDGEMENT

The authors would like to thank Dr. Denise Wilson for providing us with industrial layouts.

7. REFERENCES

[1] G. A. Allen et al. "A yield improvement technique for IC layout using local design rules," *IEEE Transactions on Computer Aided Design*, vol.11, no. 11, pp.1355-1360, Nov. 1992.

[2] C.Bamji and E. Malavasi, " Enhancement network flow algorithm for yield optimization," in *Proc. IEEE/ACM DAC*, pp. 746-751, 1996.

[3] V. K. R. Chiluvuri and I. Koren, Layout-synthesis techniques for yield enhancement ," *IEEE Transactions on Semi-conductors and manufacturing*, vol. 8:2 pp. 178-187, May 1995.

[4] I. Koren and H. C. Stapper, " Yield models for defect tolearnt VLSI circuits: A review," *in Defect and Fault tolerance on VLSI Systems*, vol. 1, I. Koren Ed. New-York: Plenum, pp.1-21, 1989.

[5] S. L. Liu and J. Allen, "Minplex: A compactor that minimizes the bounding rectangles and individual rectangles in a layout," in *Proc. IEEE/ACM DAC*, pp. 123-130, 1986.

[6] J. K. Ousterhout, "Corner stitching: A data-structuring technique for VLSI layout tools," *IEEE Transactions on Computer Aided-Design of integrated Circuits and Systems*, vol. CAD-3, no.1, pp. 87-100, January 1984.

[7] S. Sastry and A. Parker, "The complexity of two dimensional compaction of VLSI layouts," in *Proc. Int. Conf. on Circuits and Computers*, pp. 402-406, 1982.

[8] H. Xue, C. N. Di, and J. A. Jess ,"Fast multi-layer critical area computation," in *Proc. of IEEE Int. Workshop on Defect and Fault tolerance on VLSI Systems*. Oct. 1993.

[9] H. Xue, C. N. Di, and J. A. Jess,"A net-oriented method for realistic fault analysis," *in Proc. IEEE Int. Conf. On Computer Aided-Design*, pp.78-84, Nov. 1993.