Pattern Sensitive Placement For Manufacturability

Shiyan Hu
Department of ECE
Texas A&M University
College Station, Texas 77843
hushiyan@ece.tamu.edu

Jiang Hu
Department of ECE
Texas A&M University
College Station, Texas 77843
jianghu@ece.tamu.edu

ABSTRACT

When VLSI technology scales toward $45nm$, the lithography wavelength stays at $193nm$. This large gap results in strong refractive effects in lithography. Consequently, it is a huge challenge to reliably print layout features on wafers and the printing is more susceptible to lithographic process variations. Although resolution enhancement techniques can mitigate this manufacturability problem, their capabilities are overstretched by the continuous shrinking of VLSI feature size. On the other hand, the quality and robustness of lithography directly depend on layout patterns. Therefore, it becomes imperative to consider the manufacturability issue during layout design such that the burden of lithography process can be alleviated.

In this paper, the problem of cell placement considering manufacturability is studied. Instead of designing a new cell placer, our goal is to tune any existing cell placement solution to be lithography friendly. For this purpose, three algorithms are proposed, which are cell flipping algorithm, single row optimization approach and multiple row optimization approach. These algorithms are based on dynamic programming and graph theoretic approaches, and can provide different tradeoff between edge placement error (EPE) reduction and wirelength increase. Using lithography simulations, our experimental results on realistic netlists and cell library demonstrate that over 20% EPE reduction can be obtained by the new approaches while only less than 1% additional wire is introduced.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids - Placement and Routing; J.6 [**Computer-aided Engineering**]: Computer-aided Design

General Terms

Algorithms, Performance, Design

ISPD'07, March 18–21, 2007, Austin, Texas, USA.

Keywords

Manufacturability, Placement, Physical Design

1. INTRODUCTION

As VLSI technology enters the nano-scale regime, demands for minimum feature sizes have outpaced the advances in lithography hardware solutions. This imposes a great challenge on manufacturing reliability. In current lithography technology, $193nm$ wavelength is used to print $65nm$ or even $45nm$ features. This leads to a lot of refractive effects and images on wafer have remarkable mismatches from mask layouts. Lithography-induced variation also aggravates. As more variations are presented with e.g., gate length, timing and power of circuits are significantly affected.

Currently, semiconductor industry heavily relies on *resolution enhancement techniques* (RETs) for improving printability. Roughly speaking, printability refers to the difficulty in obtaining a good match between the intended image and the printed image in lithography process. Printability is often measured by *critical dimension* (CD) accuracy, which refers to the size of thin features (e.g., gate length) which are difficult to print reliably. Thus, achieving high CD accuracy means that the printed patterns well match the desired ones. Prevailing RETs for improving CD accuracy include optimal proximity correction, phase shift mask, off-axis illumination, and sub-resolution assist features [1].

RETs are effective in improving CD accuracy. However, increasingly shrinking features on the die and increasing complexity of the design over-stretch the capability of RETs. This problem aggravates when RETs are applied to the layouts which are not lithography friendly. Furthermore, RETs often complicate photomark shapes and introduce considerable amount of additional cost to photomask fabrication, which makes RETs expensive to apply. To attack the above issues, efforts are needed in all process and design stages. With respect to physical design, manufacturability-aware methodologies would be performed to reduce the burden of manufactures. Our purpose is that with more lithography-friendly layout, the tasks of manufacturers would become significantly easier and RETs become less expensive to apply.

More benefits can be obtained from design for manufacturability. In the sub-$90nm$ technology, design is heavily affected by fabrication variability. Lithography process certainly has direct impact on fabrication variability. Thus, a lithography-friendly layout has the potential to make the design more resistant to fabrication variations. As the variability has big impact on power, design for manufacturbility

also tends to obtain high quality design in terms of power.

There are some previous works related to RET-aware physical designs such as [2, 3] for routing problems. Other lithography friendly design methodologies include regular fabric [4, 5] and restricted design rules (RDR) [6]. Regular fabric methodology, which is somewhat similar to FPGA, requires circuit fabrics to be constructed from a set of regular physical geometry [4, 5]. Due to the geometric regularity, the resulting designs are RET-friendly. However, similar to FPGA based design, circuit performance is compromised. RDR imposes restrictive rules on layout designs to enhance manufacturbility [6]. However, it is difficult and expensive to use these rules to capture the non-local lithography effects. For this, many rules have to be introduced, which may intensify the problem of design rule explosion. Furthermore, RDR introduces regularity into designs which may also lead to penalty on circuit area and performance. In this paper, the problem of manufacturability-driven cell placement problem is studied, and our solutions do not have the above shortcomings. According to the best of authors' knowledge, the closest related work is [7] where a lithography-aware detailed placement approach is presented. It is to achieve high CD accuracy and thus enhance feature printability through perturbing a given detailed placement of circuits. However, it only performs spacing optimization between cells and thus does not allow changing relative locations of cells. Furthermore, it does not consider cell flipping. These turn out to be important to obtain a high-quality lithography-friendly cell placement as indicated by our experiments.

Placement of cells has remarkable effect on printability. This is due to the fact that gate lengths for transistors on the boundary regions of a cell significantly depend on its neighboring cells. Although sound library cell design can achieve high printability for internal transistors, it cannot handle the boundary transistors. On the other hand, as the gate length keeps shrinking with technologies, the placement will affect deeper and deeper regions of the cells.

In this paper, several manufacturability-driven new cell placement algorithms are proposed. Our goal is to modify any existing cell placement solution to make it lithography friendly. The new methods start with a placement obtained from any existing placer, and improve CD accuracy through postprocessing optimizations. Precisely, the location and the orientation of each cell can be perturbed to achieve a lithography-friendly design. As the initial placement is computed by salient (non-lithography-driven) CAD tools and thus of high quality (in terms of e.g., wirelength), it is desired to limit the perturbation to it when improving CD accuracy. Thus, our problem is to compute a lithography-friendly layout subject to perturbation constraints. For this purpose, three algorithms are proposed which are dynamic programming based cell flipping algorithm, single row based optimization approach and multiple row based optimization approach. To measure printability, *edge placement error*, or EPE in short, is used. To measure perturbation, wirelength increase is used. Our experimental results demonstrate that over 20% EPE reduction can be obtained by the new approaches while only less than 1% additional wire is needed.

The rest of the paper is organized as follows: Section 2 formulates the perturbation constrained lithography-driven cell placement problem. Section 3 describes the cell flipping algorithm. Section 4 describes single row optimization approach and multiple row optimization approach. Section 5 presents the experimental results with analysis. A summary of work is given in Section 6.

2. PRELIMINARIES

2.1 Motivation

As demands for minimum feature sizes have outpaced the advances in lithography hardware solutions, there may be considerable amount of distortions between the intended image and the actual printed image in the lithography process. Printability refers to the different levels of distortions. Given an initial cell placement, our goal is to achieve high printability through performing postprocessing optimizations to it.

It is helpful to see a simple example which demonstrate that cell placements can affect the printability. Refer to Figure 1 for such an example. Figure 1(a) shows a placement of three gates obtained using a realistic $130nm$ cell library. Figure 1(b) shows another placement obtained by flipping the middle NOR gate. In this way, we immediately obtain a much more lithography-friendly layout.

2.2 Problem Formulation

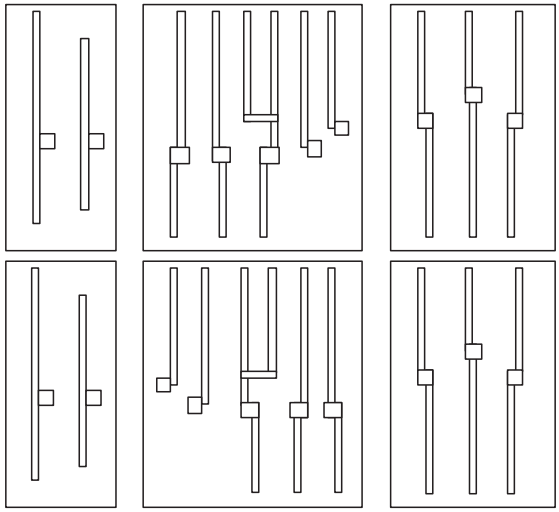

Figure 1: Lithography optimization through cell flipping. This design is extracted from an ISCAS'89 benchmark circuit, where a buffer, a NOR gate and a NAND gate are placed in series. Though flipping the middle NOR gate, the manufacturability cost (which is measured by total edge placement errors in this example) is reduced from 0.09 to 0.05. Gate length variation is reduced from 21% of the nominal value to 11% of the nominal value. Rectangles shown are polys.

Distortions in lithography process can be measured by a generic function *pattern dependent manufacturability cost*, or *manufacturability cost* in short, denoted by $\eta(\cdot)$. $\eta(\cdot)$ is defined on *pattern* which is associated with cells, i.e., a pattern could be cells or part of cells. In this paper, we are interested in the pattern spanning only two horizontally adjacent cells. Denote by C a cell, by C^l the left side of C, and by C^r the right side of C. If a pair of cells C_i, C_j are adjacently placed in a row, a pattern $P(C_i^r, C_j^l)$ associated to

them could refer to the part spanning over C_i^r and C_j^l. For each pattern P, we have a manufacturability cost $\eta(P)$. This manufacturability cost can refer to many instances such as different instances: edge placement error (EPE), image log slope (ILS), process window, lithography-induced variation, etc [1]. It is important to note the following facts:

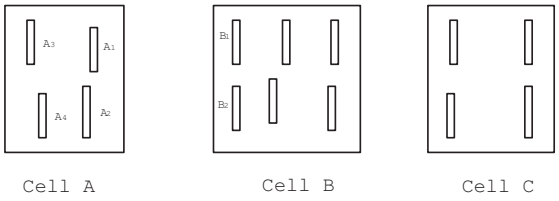

Figure 2: Definition of EPE cost for cells. Rectangles shown are polys.

- As a cell is associated with an orientation, when the orientation is changed (i.e., cell flipping happens), any pattern associated to this cell is in general also changed. For example, in Figure 2, η for the pattern between A, B depends on A_1, A_2, B_1, B_2. When cell A is flipped, η for the pattern between A, B will depend on A_3, A_4, B_1, B_2

- In lithography process, adjacent cells with different spacing can have different printability. This is automatically handled using patterns as the same cell pair with different spacing will be treated as different patterns.

- Since the printability of a cell very weakly depends on cells at other rows [1], it is safe to neglect it. Formally, $\eta(P) = 0$ for any P associated with cells in different rows. This is why we are only interested in pattern associated with horizontally adjacent cells.

Given an initial cell placement, our goal is to reduce the manufacturability cost η through postprocessing optimizations. For this, all patterns in the placement are investigated, and those which are not lithography-friendly, i.e., tend to reduce functional and parametric yield, are identified. Optimizations are then performed there to make the layout lithography-friendly. Thus, our lithography-driven optimization is *pattern sensitive*.

Online evaluation of manufacturability cost for each pattern is time consuming and not necessary. A better idea is to compute $\eta(P)$ for each possible pattern P off-line and store them in a lookup table for future usage. As P is associated with adjacent cells, thus cell orientations and spacing between cells need to be considered when building the lookup table. The following benefits can be obtained due to lookup table bases lithography optimizations:

- Online lithography simulations, which are very computationally expensive, are avoided. This is a key difference between our approach and those in [2, 3]. There exists fast lithography simulations (e.g., the one in [3]). However, as many runs of online simulations have to be performed during circuit optimizations, such approaches can still be improved. As performing a simulation is much slower than obtaining a number in a lookup table, one could use lookup table based optimization for speedup.

- As the lookup table is built off-line, full-fledged expensive lithography simulations can be performed. Compared to [3] where a fast aerial image (which is a first order approximation to the optimal system) simulation is performed, we estimate printability using more accurate approximations.

As manufacturability cost between cells in different rows is negligible, row-based placement approaches are designed in this paper. For this, we define *row η cost* for a row of cells as the sum of η between all adjacently placed cell pairs in the row, and define the *total η cost* for the whole placement as the sum of all row η costs. In this paper, we propose to adjust the placement to reduce the total η cost. Such adjustment should be as slight as possible so as to introduce minimal amount of perturbation to the design. This is desired as the initial design, although not lithography-friendly, should have high quality in terms of e.g., wirelength as it is returned by salient (non-lithography-driven) CAD tools.

Three types of adjustments are considered in this paper. The first type of adjustments is not to change the locations of cells, rather, it only allows changing orientations of cells. We call it *Cell Flipping Optimization*. Refer to Figure 1 for an example illustrating the impact of cell flipping on η. Evidently, the manufacturability cost is significantly reduced in this example. Note that cell flipping optimization has also been used in [8] for wirelength reduction in cell placement. The second type of adjustments allows both cell re-location and cell flipping. To introduce small amount of perturbation to the original placement, each cell is only allowed to move within a small range around its original location. Furthermore, when a row is adjusted, all other rows must be fixed. Thus, this type of adjustments is called *Single Row Optimization*. The third type of adjustments is the same as single row optimization except that several neighboring rows are optimized simultaneously, that is, when a row is adjusted, it neighboring rows are adjusted as well. This type of adjustments is called *Multiple Row Optimization*. Clearly, increasing amount of efforts are needed in these three optimizations, and one may expect that increasing amount of reduction in manufacturability cost can be obtained. Our problem is formulated as follows.

Perturbation Constrained Lithography-Driven Cell Placement Problem: Given a cell placement, we are to perform post-processing optimizations, which can be cell flipping, single row optimizations or multiple row optimizations, such that the total manufacturability cost (i.e., total η cost) is reduced subject to the constraint α on perturbation.

In this paper, we measure the *perturbation* to a placement by wirelength increase as wirelength has direct relationship with timing and is efficient to compute. Thus, the perturbation constraint α actually refers to the maximum tolerable wirelength increase ratio. Note that other similar metrics could be easily incorporated into our new approach.

Finally, although this paper is restricted to row-based

postplacement designs, the ideas can be extended to handle non-row-based designs. In particular, Multiple Row Optimization approach can be directly applied to other placement styles.

3. CELL FLIPPING

The first algorithm, namely, cell flipping algorithm, works under the dynamic programming framework. As our cell placement algorithm is a row-based approach, cell flipping algorithm is carried out row by row. In a row, the location of each cell is fixed and the orientation of each cell is to be determined. For convenience, "cells" in this section simply refer to a row of cells.

We define a *partial* cell flipping solution to be an incomplete determination for the orientations of all cells. A partial solution becomes *complete* when the orientations of all cells are determined. A cell is *processed* if its orientation has been determined.

3.1 Algorithmic Overview

Given an initial cell placement, cells are first sorted in the topological order (i.e., from left to right). We then start from the first cell and set its orientation to each of two possible choices (i.e., flipped or un-flipped), which results in two partial solutions. For each partial solution, we process the second cell and set its orientation to each of two choices. In this way, the algorithm proceeds in a dynamic programming fashion, i.e., it processes each cell in turn according to the topological order. Without any solution pruning, there are certainly 2^n solutions for optimizing n cells. Therefore, during the solution propagation process, inferior solutions are pruned for acceleration. The algorithm terminates when all partial solutions become complete and the solution with the minimal η cost and satisfying wirelength constraint is returned.

3.2 Solution Characterization

A set of partial solutions $\mathcal{S}$ keep being updated during the process of dynamic programming. Each solution $S \in \mathcal{S}$ is associated with a (CE, CW) pair, where CE denotes the cumulative η cost for all processed cells, and CW denotes the cumulative wirelength. Note that CW is computed using the widely-used metric *half-perimeter wirelength* (HPWL) on all those nets which do not span on any unprocessed cell.

3.3 Solution Propagation

Suppose that a cell C is "inserted" to the current partial solution S, i.e., we are to decide the orientation of C. A new solution S' will be formed for each possible cell insertion (flipped C and unflipped C). Because all cells are processed according to the topological order, when C is processed, the cumulative η cost can be updated by

$$CE(S') = CE(S) + \eta(P(C^r_{last}, C^l)),$$

where C_{last} is the last processed cell. The cumulative wirelength is updated by recomputing HPWL of all nets not spanning on any unprocessed cell.

3.4 Solution Pruning

During the process of dynamic programming, there may be a lot of solutions. Some of them are inferior to others and they will be pruned to accelerate the approach.

For any two solutions S_1, S_2 *with the same set of processed cells*, S_2 is *inferior to* S_1 if $CE(S_2) \geq CE(S_1)$ and $CW(S_2) \geq CW(S_1)$. That is, a solution is inferior to another if it has worse cumulative η cost and worse cumulative wirelength. Whenever a solution becomes inferior, it is pruned from the solution set without further propagation. A solution S can also be pruned when it is infeasible, i.e., its cumulative wirelength is greater than the wirelength constraint of that row. For a row, the wirelength constraint is set to $(1+\alpha)\cdot L$, where L is the total wirelength of the row in the original (i.e., initial) placement and α is the maximum tolerable wirelength increase ratio.

4. SINGLE ROW OPTIMIZATION AND MULTIPLE ROW OPTIMIZATION

4.1 Algorithmic Overview (Single Row Optimization)

In single row based optimization, in addition to the cell orientation, we are allowed to change the location of each cell. In this way, more η reduction is expected. Since small perturbation is desired, each cell is only allowed to be movable within a small range around its original position. To approximately implement this strategy, cells will be processed by groups. Every consecutive k cells form a *group* and optimizations (including relocation and cell flipping) are performed inside each group. At any time, only one group is optimized. To determine which group to be optimized, a multi-dimensional descent based optimization approach is used. Such an approach has been successfully used in CAD problems including [9] for gate sizing.

At the beginning, a set of groups called *improvablegroups* are formed as follows. Each group in the row will be assigned with a *cost* which is equal to the possible η reduction for this group (computed by tentatively optimizing the group) when all other cells are fixed. As long as the cost for a group is positive (i.e., η reduction is possible), the group is included into improvablegroups. A subset of improvablegroups, called *optimizedgroups*, are then computed as an ordered set of groups which may provide cumulative improvement in cost. Optimizations are then performed to each group in optimizedgroups in turn. Subsequently, improvablegroups are set to optimizedgroups and the above process is repeated until convergence.

The remaining question is how to compute η reduction for a group of cells. Precisely, our goal is to compute a new cell placement (for this group of cells) with reduced η cost subject to the constraint on wirelength increase. For this, we will first compute the "best η" solution, i.e., the optimal η-driven placement solution without considering wirelength constraint. Since the original placement is returned by salient CAD tools, it is reasonable to treat it as the "best wire" solution. Subsequently, an iterative approach is performed to gradually turn the best η solution into the best wire solution. The process terminates when wirelength increase ratio satisfies the constraint α (together with non-overlapping requirement). Since our goal is to obtain a minimum η solution subject to the wirelength constraint, it makes sense to gradually modify the best η to obtain the solution satisfying the wirelength constraint and still with good η. The approach will be detailed in Section 4.2 and Section 4.3.

4.2 Unconstrained Optimal Manufacturability-Driven Placement

A critical observation is that when wirelength is not considered, for a group of cells, the cell placement achieving the minimal η cost can be obtained through reduction to the *minimum cost Hamiltonian path problem*.

A graph $G = (V, E)$ is to be constructed as follows. Each cell C in the group will be mapped to two nodes $v_{C,l}$ and $v_{C,r}$ in G, where $v_{C,l}$ corresponds to the left side of unflipped C and $v_{C,r}$ corresponds to the right side of unflipped C. There is an edge between $v_{C,l}$ and $v_{C,r}$ with weight 0. For any two nodes v_i and v_j which belong to different cells, there is an edge $v_i v_j$. The weight associated with such an edge is equal to the smallest η cost between v_i and v_j when placing v_i (on the left) and v_j (on the right) adjacently, i.e., $P(v_i, v_j)$. Note that η cost function is in general not monotonic with whitespace (see, e.g., the case of edge placement error [7]), and the smallest η can be obtained by our η lookup table. Refer to Figure 3 for the graph corresponding to Figure 2. Note that weights for edges $v_{A,l}v_{A,r}, v_{B,l}v_{B,r}, v_{C,l}v_{C,r}$ are all 0. As an example, we show how to compute the weight of edge $v_{A,l}v_{B,r}$. Since the weight of an edge refers to the η for the part between the two nodes, we have to first flip cell A and cell B to make the left side of cell A (corresponding to $v_{A,l}$) directly connect to the right side of cell B (corresponding to $v_{B,r}$). Denote the flipped C by C_f. Thus, the weight for $v_{A,l}v_{B,r}$ is equal to $\eta(P(C^l_{A,f}, C^r_{B,f}))$.

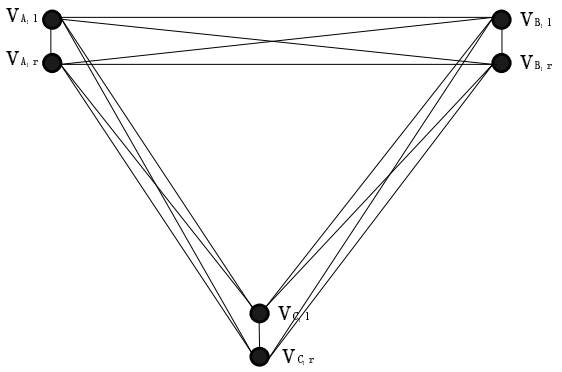

Figure 3: Graph G corresponding to Figure 2.

To compute the optimal η-driven cell placement (i.e., best η solution) for this group of cells, it suffices to compute a path visiting each node exactly once such that the total edge weights along the path is minimized. This problem is the *Minimum Cost Hamiltonian Path Problem*. For example, in Figure 3, if $v_{B,l}v_{B,r}v_{A,r}v_{A,l}v_{C,l}v_{C,r}$ is returned as the minimum cost Hamiltonian path, then we know that in best η solution, we need to place B, A, C in this order and B, C are unflipped while A is flipped.

As the minimum cost Hamiltonian path problem is an NP-complete problem [10], the following closest-point heuristic (which is similar to the one in [10]) is used to compute the efficient approximation. Define *a partial Hamiltonian path* to be an incomplete Hamiltonian path. The algorithm begins with picking an arbitrary node in G. At each step, the node which is not yet included in the partial Hamiltonian path and is closest to any point along the partial Hamiltonian path is identified. Denote this node by u and suppose that it is closest to v along the partial path. We will insert u to the path just after v. Note that whenever a node is included, another node belonging to the same cell (i.e., the node corresponding to another side of the cell) must also be included. For example, if $v_{A,r}$ is picked for insertion, then $v_{A,l}$ must be inserted immediately after $v_{A,r}$. In this way, we guarantee that the resulting path is valid for placement. It is clear that the close-point heuristic has this property.

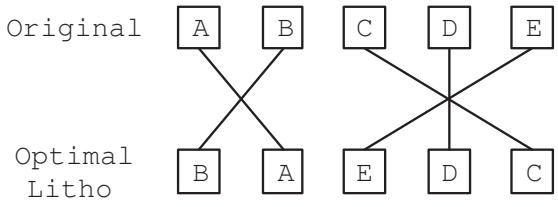

Figure 4: Obtaining tradeoff between η-cost and wirelength.

4.3 Wirelength Constrained Manufacturability-Driven Placement

When wirelength is not considered, optimal η-driven placement can be computed as in Section 4.2. This subsection deals with turning an unconstrained solution into a wirelength constrained solution. Our idea is to start from the best η solution and perform local adjustment to make its wirelength increase fall into the requirement. Precisely, our local adjustment is an iterative approach and it gradually turns the cell placement to be closer and closer to the original placement until the wirelength increase satisfies the constraint. Since η cost and wirelength depend on the spacing (i.e., whitespace) between cells, our approach also performs spacing optimization.

Let us illustrate the approach using a simple example. Suppose that the original cell placement and optimal η-driven cell placement are as shown in Figure 4. Our algorithm first identifies an ordered set of cell pairs for location exchange to turn the optimal η-driven placement (best η solution) into the original cell placement (best wire solution). For this, an iterative procedure is used. At each iteration, a pair of cells are identified for location exchange. A cell is first linked to its target location (which is the location in best wire placement) as in Figure 4. A cell which is not at its target location and is with maximum link crossings is then identified[1]. Another cell to be exchanged with it is the one at the target location of the identified cell. For example, in Figure 4, cell E is first identified and it is to exchange location with cell C. It is easy to see that the ordered set of exchange cell pairs are $\{EC, BA\}$.

After identifying the ordered set of cell pairs, we are to perform cell location exchange one by one. After each location exchange, spacings between cells are inherited from the ones before location exchange with the following exception. For those placement patterns of cell pairs which can be found in the best wire placement, spacings between them are set as in the best wire placement. For example, after E, C are exchanged in Figure 4, spacing between CD and DE will be set as in original placement and spacing between AC is inherited, i.e., it is equal to that of AE in the optimal η solution. Note that since our optimizations are performed

[1]In case of a tie, an arbitrary cell in tie is identified.

within a group, we need to guarantee that after optimization, placement of the group will not overlap with its neighboring groups. Thus, location exchange process terminates when wirelength increase ratio of the placement falls below α and the placement does not overlap with its neighboring groups. In this way, we can obtain a good η-driven cell placement subject to the wirelength constraint.

4.4 Extension to Multiple Row Optimization

Suppose that there are five rows of cells to be placed. By single row optimization, these five rows will be optimized separately, i.e., the first row of cells will be optimized followed by the second row and so forth. However, when a row is optimized, it would be possible that its neighboring rows need to be accordingly modified to achieve an overall good design. This is due to the fact that a net often spans a few neighboring rows, and thus wirelength could be reduced with adjusting several rows simultaneously. As a consequence, some previously "infeasible" placements (i.e., the one violating wirelength constraint) which provide large amount of η reduction may become feasible. This may eventually leads to that the solution with more η reduction is returned. This motivates multiple row optimization.

The algorithm for single row optimization can be readily extended to handle multiple row optimization. In multiple row optimization, m neighboring rows will be grouped to form a *row group* and optimizations are performed within each row group. The algorithm is as follows.

For each row within a row group, cells are grouped as in single row optimization. All resulting groups (in all rows of the row group) are put into a single set called *group set*. Our purpose is to treat groups from different rows in the same way for "simultaneous optimization". It is true that at any time, only one group can be optimized. However, viewing at the level of group set optimization, groups of different rows can be optimized interactively. For example, it is possible that optimizing a group at a row may impact some groups at other rows and performing successive optimizations there may be very beneficial. Our multiple row optimization captures this and interactions among neighboring rows are explored. In this sense, multiple rows are optimized "simultaneously". Since any group in the group set must be located in a single row, approaches for computing wirelength constrained η-driven placement as described in Section 4.2 and Section 4.3 can be directly applied. The multi-dimensional descent based approach is also readily applied. Precisely, we pick a candidate set of groups (which may be in different rows) as improvablegroups and identify a subset of it which can improve cost. After performing optimizations to this subset, improvablegroups is updated and this process is repeated until convergence. Note that since our optimization is performed into a group whose cells must be in the same row, no cell can move to any other row after optimization.

5. EXPERIMENTAL RESULTS

5.1 Experiment setup

The algorithms of Cell Flipping, Single Row Optimization and Multiple Row Optimization are implemented in C++ and are tested on a Pentium IV computer with a 3.0GHz CPU and 2G memory. Two sets of testcases are used in experiments, namely, ISPD'04 benchmark circuits and ISCAS'89 benchmark circuits. In performing lithography-driven postprocessing optimizations, the perturbation constraint, i.e., the maximum tolerable wirelength increase ratio α, is set to 1%.

Similar to previous works [3, 7], manufacturing distortions are measured by *edge placement error* (EPE), i.e., η is realized using EPE in this paper. Precisely, EPE refers to the difference between the placements of edges of lines as well as other features in the printed image and those in the intended image.

5.2 Experiments with ISCAS'89 benchmark circuits

We first perform experiments on ISCAS'89 benchmark circuits. Logical synthesis and technology mapping (using Berkeley SIS) with a realistic cell library for $130nm$ technology, which consists of 22 cells, are performed to the circuits. Our lookup table for EPE cost is built as follows. 10 representative whitespace between adjacent cells is considered. They are $1\nu, 3\nu, \ldots, 19\nu$ where $\nu = 55nm$. For each pair of cell types, for each possible cell orientation, and for any of 10 representative whitespace, an EPE is obtained by SPLAT [11]. As an analogue to SPICE for circuit simulation, a lithography simulator called *Simulation of Projection Lens Aberrations via Transmission Cross-Coefficients* (SPLAT) [11] is used. Initial placements are computed using FastPlace [12]. Note that other placers can certainly be used. Placements are then optimized for EPE reduction and the results are summarized in Table 1. We make the following observations.

- For Cell Flipping algorithm, on average about 9% EPE reduction is obtained with 0.17% additional wire. Cell Flipping algorithm runs fastest among all algorithms, which makes sense as the smallest amount of effort is needed there.
- For Single Row Optimization algorithm, on average 14.6% EPE reduction is obtained, which improves the results by Cell Flipping. The amount of additional wire is still small. On average, only 0.35% more wire is needed.
- For Multiple Row Optimization algorithm, on average more than 22% EPE reduction is obtained, which gives the best results among all three algorithms. At the same time, it needs more wire. Multiple Row Optimization runs slowest among all three algorithms.

Our optimization approaches can be tuned to obtain different tradeoff between EPE reduction and wirelength. For example, in Single Row Optimization, one can impose a maximum number of iterations to the implementation so as to terminate the multi-dimensional descent based optimization procedure before it converges. By varying this number, different tradeoff can be obtained. As an example, a tradeoff curve is shown in Figure 5.

5.3 Experiments with ISPD'04 benchmark circuits

We next perform experiments on a standard placement benchmark, namely, ISPD'04 benchmark circuits [12, 13]. The statistics of the circuits are shown in Table 2. The circuits are first placed to obtain initial placement results using

Table 1: Performance of each algorithm on ISCAS'89 Benchmark Circuits.

Circuit		Cell Flipping			Single Row Optimization			Multiple Row Optimization		
Name	#Nodes	Wire Inc.	EPE Red.	CPU (s)	Wire Inc.	EPE Red.	CPU (s)	Wire Inc.	EPE Red.	CPU (s)
s838	317	0.05%	5.5%	0.1	0.09%	9.2%	0.1	0.22%	18.5%	0.2
s1238	439	0.11%	5.1%	0.1	0.25%	9.0%	0.2	0.50%	18.9%	0.3
s1423	511	0.25%	9.1%	0.1	0.33%	15.3%	0.2	0.51%	25.3%	0.4
s1488	429	0.15%	6.5%	0.1	0.39%	12.8%	0.3	0.81%	19.5%	0.9
s5378	1227	0.37%	12.1%	0.5	0.52%	13.0%	0.7	0.73%	22.7%	1.1
s9234	1162	0.03%	12.7%	0.5	0.12%	18.7%	0.9	0.39%	23.3%	1.3
s15850	3621	0.12%	10.2%	2.1	0.56%	19.1%	4.1	0.83%	23.6%	5.7
s35932	13460	0.30%	9.9%	18.9	0.77%	13.0%	41.7	0.95%	24.0%	95.2
s38417	8965	0.22%	8.2%	7.7	0.29%	16.7%	17.8	0.91%	21.2%	32.5
s38584	10463	0.05%	11.3%	16.4	0.15%	18.8%	28.2	0.29%	23.5%	38.7
Average	4059	0.17%	9.1%	4.7	0.35%	14.6%	9.5	0.57%	22.1%	17.6

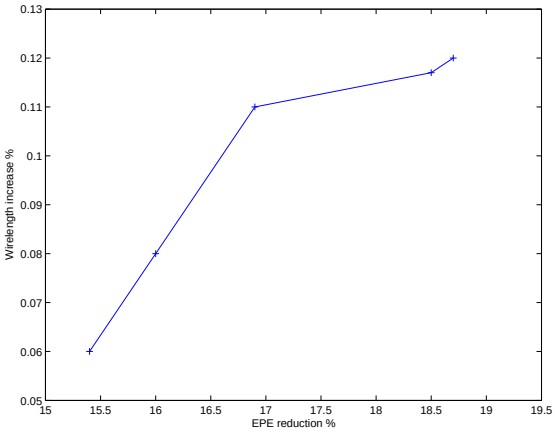

Figure 5: Tradeoff between EPE reduction and wirelength increase using Single Row Optimization for s9234.

FastPlace [12]. They are then optimized for EPE reduction and the results are summarized in Table 3. Since the original circuits (including e.g., gate type of each node) for ISPD'04 benchmark are not known to us, our EPE lookup table is randomly generated in this case. We make the following observations.

- For Cell Flipping algorithm, on average more than 11% EPE reduction is obtained, with only 0.16% additional wire. In the worst case, only 0.32% additional wire is needed.

- For Single Row Optimization algorithm, often more than 18% EPE reduction is obtained, which improves the results by Cell Flipping. The amount of additional wire is still small. On average, only 0.29% more wire is needed.

- For Multiple Row Optimization algorithm, on average 25% EPE reduction is obtained, which gives the best results among all three algorithms. At the same time, it needs 0.41% additional wire. As it spends a lot of effort on optimizations, it runs slowest among all algorithms.

Table 2: Statistics of ISPD'04 benchmark circuits [12, 13].

Circuit	#Cells	#Pads	#Nets	#Pins	#Rows
IBM01	12506	246	14111	50566	96
IBM02	19342	259	19584	81199	109
IBM03	22853	283	27401	93573	121
IBM04	27220	287	31970	105859	136
IBM05	28146	1201	28446	126308	139
IBM06	32332	166	34826	128182	126
IBM07	45639	287	48117	175639	166
IBM08	51023	286	50513	204890	170
IBM09	53110	285	60902	222088	183
IBM10	68685	744	75196	297567	234
IBM11	70152	406	81454	280786	208
IBM12	70439	637	77240	317760	242
IBM13	83709	490	99666	357075	224
IBM14	147088	517	152772	546816	305
IBM15	161187	383	186608	715823	303
IBM16	182980	504	190048	778823	347
IBM17	184752	743	189581	860036	379
IBM18	210341	272	201920	819697	361

6. CONCLUSION

Traditionally, design and manufacturing process are separate. This trend should be turned so as to make the resolution enhancement techniques easy and less expensive to apply. In this paper, several new algorithms are proposed for manufacturability-driven cell placement. They are cell flipping algorithm, single row based optimization and multiple row based optimization approaches. Cell flipping algorithm works under the dynamic programming framework. In row-based optimizations, cells are partitioned into groups and are optimized through reduction to graph theoretic problems such as the minimum cost Hamiltonian path problem. These algorithms are very effective in reducing EPE cost and are able to provide different tradeoff between EPE reduction and wirelength increase. Our experimental results demonstrate that $> 20\%$ EPE reduction can be obtained by the new approaches with only $< 1\%$ additional wire. Although this paper is restricted to row-based designs, the ideas can be extended to handle non-row-based designs. In particular, Multiple Row Optimization approach can be directly applied to other placement styles.

Table 3: Performance of each algorithm on ISPD'04 Benchmark Circuits.

	Cell Flipping			Single Row Optimization			Multiple Row Optimization		
Circuit	Wire Inc.	EPE Red.	CPU (s)	Wire Inc.	EPE Red.	CPU (s)	Wire Inc.	EPE Red.	CPU (s)
IBM01	0.31%	10.8%	12.3	0.53%	18.7%	28.2	0.75%	31.3%	48.9
IBM02	0.09%	11.4%	29.5	0.13%	21.5%	72.9	0.23%	28.1%	124.5
IBM03	0.32%	10.7%	49.7	0.53%	16.8%	96.3	0.69%	22.0%	190.2
IBM04	0.22%	10.3%	47.3	0.31%	19.1%	143.3	0.50%	26.2%	245.8
IBM05	0.08%	14.8%	62.5	0.12%	19.8%	172.6	0.70%	26.5%	301.4
IBM06	0.13%	9.2%	73.5	0.23%	20.9%	246.1	0.47%	29.3%	453.3
IBM07	0.02%	12.5%	154.7	0.05%	17.3%	429.7	0.11%	26.7%	878.1
IBM08	0.25%	12.8%	247.5	0.17%	17.5%	480.0	0.37%	25.7%	1236.7
IBM09	0.27%	9.6%	345.8	0.41%	15.6%	754.2	0.59%	23.0%	1362.5
IBM10	0.16%	11.5%	431.7	0.17%	16.0%	894.7	0.25%	20.4%	1437.9
IBM11	0.29%	12.9%	625.1	0.57%	21.8%	1175.3	0.79%	24.1%	1598.1
IBM12	0.07%	9.6%	755.4	0.39%	15.1%	1593.9	0.58%	20.6%	1831.8
IBM13	0.22%	10.9%	1311.5	0.38%	20.6%	1750.8	0.49%	25.0%	2263.0
IBM14	0.08%	13.0%	1899.1	0.15%	19.9%	3065.2	0.18%	22.8%	3910.1
IBM15	0.16%	9.6%	2058.8	0.25%	17.9%	3375.2	0.29%	23.1%	4087.9
IBM16	0.07%	11.5%	3036.4	0.08%	14.9%	3512.5	0.15%	24.3%	4336.4
IBM17	0.04%	12.8%	3579.1	0.09%	21.2%	4371.1	0.13%	28.5%	5501.5
IBM18	0.03%	9.5%	2622.0	0.06%	15.4%	3975.8	0.10%	21.3%	4393.9
Average	0.16%	11.3%	963.5	0.29%	18.3%	1453.8	0.41%	25.0%	1900.2

7. REFERENCES

[1] A.-K. Wong, "Resolution enhancement techniques in optical lithography," *SPIE Press*, 2001.

[2] L.-D. Huang and D. Wong, "Optical proximity correction (opc)-friendly maze routing," *Proceedings of ACM/IEEE Design Automation Conference*, pp. 186 – 191, 2004.

[3] J. Mitra, P. Yu, and D. Pan, "Radar: Ret-aware detailed routing using fast lithography simulations," *Proceedings of ACM/IEEE Design Automation Conference*, pp. 369 – 372, 2005.

[4] V. Kheterpal, T. Hersan, V. Rovner, D. Motiani, Y. Takagawa, L. Pileggi, and A. Strojwas, "Design methodology for ic manufacturability based on regular logic-bricks," *Proceedings of ACM/IEEE Design Automation Conference*, 2005.

[5] L. Pileggi, H. Schmit, A. Strojwas, P. Gopalakrishnan, V. Kheterpal, A. Koorapaty, C. Patel, V. Rovner, and K. Tong, "Exploring regular fabrics to optimize the performance-cost trade-off," *Proceedings of ACM/IEEE Design Automation Conference*, 2003.

[6] L. Liebmann, "Layout impact of resolution enhancement techniques: impediment or opportunity?" *Proceedings of International Symposium on Physical Design*, pp. 110–117, 2003.

[7] P. Gupta, A. Kahng, and C.-H. Park, "Detailed placement for improved depth of focus and cd control," *Proceedings of Asia and South Pacific Design Automation Conference*, pp. 343–348, 2005.

[8] C.-W. Sham, E. Young, and C. Chu, "Optimal cell flipping in placement and floorplanning," *Proceedings of ACM/IEEE Design Automation Conference*, pp. 1109 – 1114, 2006.

[9] O. Coudert, "Gate sizing for constrained delay/power/area optimization," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 5, no. 4, pp. 465–472, 1997.

[10] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. MIT Press, 2001.

[11] http://cuervo.eecs.berkeley.edu/Volcano/docs/splat/.

[12] N. Viswanathan and C. C.-N. Chu, "Fastplace: Efficient analytical placement using cell shifting, iterative local refinement and a hybrid net model," *Proceedings of International Symposium on Physical Design*, pp. 26–33, 2004.

[13] http://www.public.iastate.edu/~nataraj/ISPD04_Bench.html.