Dummy Fill Optimization for Enhanced Manufacturability *

Yaoguang Wei and Sachin S. Sapatnekar
Department of Electrical and Computer Engineering
University of Minnesota, Minneapolis, MN 55455
weiyg, sachin@umn.edu

ABSTRACT

This paper presents a router that minimizes the amount of dummy fill necessary to satisfy the requirements for chemical-mechanical polishing (CMP). The algorithm uses a greedy strategy and effective cost functions to control the maximal effective pattern density during routing. On a standard set of benchmark circuits, our CMP-aware router can reduce the required dummy fill by 22.0% on average, and up to 41.5%, as compared to the CMP-unaware case. In comparison with another CMP-aware routing approach, our algorithm is demonstrated to reduce the amount of dummy fill by 14.1% on average, and up to 23.6%, over the benchmarks.

Categories and Subject Descriptors

B.7.2 [**Hardware**]: Integrated Circuit—*Design Aids*

General Terms

Algorithms, Design, Performance

1. INTRODUCTION

In nanometer-scale integrated circuit technologies, planarization steps in the manufacturing process represent a significant potential source of yield loss. An important step in planarization is chemical mechanical polishing (CMP). For example, oxide CMP is used to polish the interlayer dielectric layer (ILD) to ensure a near-planar surface before depositing and patterning a metal layer. If significant surface topography variations are seen after this process, then the depth of focus in lithography is affected, which in turn leads to variations in the critical dimension [10], resulting in performance degradation and yield loss.

In order to improve the quality of CMP, in addition to the metalized interconnects that serve an electrical function, dummy features are typically added to the layout to control the variation in the post-CMP topology [9]. Dummy feature, also referred to as dummy fill, may either be connected to power/ground (tied fill) or left floating (floating fill), and lead to increased parasitic capacitance in the layout. Floating fill increases the coupling capacitance uncertainty and can lead to signal-integrity issues, while tied fill reduces this problem, but result in high routing costs, increasing the likelihood of requiring engineering change orders [10]. For all these reasons, it is desirable to reduce the volume of dummy fill inserted into a layout.

It is known that the post-CMP ILD thickness is linearly determined by the effective pattern density (EPD) of the layout [14]. The EPD is formally defined in Section 2, but coarsely speaking, the EPD at a given location is a weighted average of the wire density in its neighborhood. If the EPD can be made more even throughout the layout, the variation of the ILD thickness after CMP can be reduced, leading to a reduction in the amount of dummy fill. Since routing plays a very major role in deciding the spatial distribution of wires in the layout, a good way to achieve greater uniformity in the EPD is to leverage the router in ensuring that the density after routing is as uniform as possible, while meeting other traditional routing objectives.

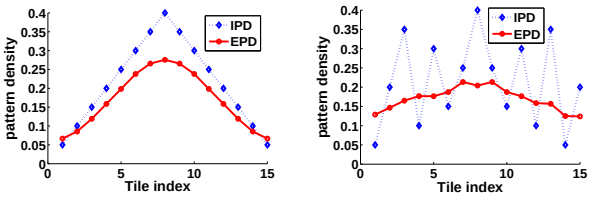
Recently, several routing algorithms considering wire distribution have been proposed in the literature. The algorithm in [12] is among the first to incorporate CMP variation into a routing algorithm, and it proceeds by attempting to balance the initial pattern density (IPD) to decrease CMP variation. The IPD is formally defined in Section 2, and is essentially the wire density. A global routing algorithm considering Cu CMP variation is proposed in [3], using an empirically developed predictive CMP density model from industry. As pointed out in [2], the algorithms in [3, 12] attempt to optimize CMP variation by only considering the IPD *inside* a routing tile/grid, which is not a right metric for CMP control, since the topographic variation is a long range effect that is affected by the IPD in neighboring tiles. In [2] a multilevel routing algorithm for oxide CMP variation is presented, using the IPD gradient as an optimization objective. However, this objective also suffers from significant limitations, as demonstrated by an example in Fig. 1, which shows two different layouts of an illustrative circuit. Layout II shows larger IPD gradients, but due to averaging effects (the size of weighting window, defined in Section 2, is 5 tiles for each layout), the EPD/CMP variation is smaller. This indicates that the gradient of the IPD may not be a good metric in the optimization of CMP variation. This claim will be demonstrated experimentally in Section 6. In essence, the methods in [2, 3, 12] attempt to decrease the variation of CMP only according to the IPD in a tile and its immediate neighbors. However, metrics such as the IPD and the IPD gradient are only indirect measures of the EPD, and their ability to optimize CMP variation is likely to be inferior to an approach that addresses the EPD variation more directly.

More recently, two routing algorithms considering EPD optimization directly have been presented [8, 17]. In [8, 17], the EPD is taken as part of the cost of a routing tile directly. While this is better than using the IPD in the cost function, the approach only considers the EPD *inside* a tile t_i as a route passes through t_i, but the impact of this route on the EPD of neighboring tiles is not considered in the cost function. Moreover, the amount of dummy fill is not directly optimized as an objective of routing. These factors limit the effectiveness of the optimization.

This paper proposes a global routing algorithm that incorporates the optimization of oxide CMP variation, in addition to the usual routing objectives. Our goal is to minimize the required dummy

*This work was supported in part by the SRC under contract 2007-TJ-1572.

ISPD'10, March 14–17, 2010, San Francisco, California, USA.



(a) Layout I: IPD gradient $\leq$ 0.05. EPD variation = 0.209.

(b) Layout II: IPD gradient $\geq$ 0.10. EPD variation = 0.089.

Figure 1: IPD and EPD topographies for Layout I and II. Layout I has smaller IPD gradients, but larger EPD variation.

fill under an accurate oxide CMP model [14], and we demonstrate, both theoretically and through experiments, that a good surrogate for this objective is to minimize the maximum EPD during routing. We elaborate cost functions to achieve this goal, and build a router that attempts to minimize the maximal EPD in routing. Our router is based on NTHU-Route 2.0 [1], which will be introduced in Section 3. Experimental results demonstrate that our algorithm can reduce the dummy fill substantially.

2. PRELIMINARIES

The goal of our global router is to optimize the overflow, wire length, and the amount of dummy fill, D, inserted for CMP planarization. Like most global routing approaches, our approach tessellates the chip into $n_r \times n_c$ grids, and constructs the global routing graph (GRG), $G = (V, E)$. Each node in V represents a grid in the layout, and an edge in E denotes the boundary between two adjacent grids.

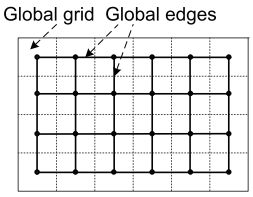


Figure 2: Global routing graph.

An example of the GRG is shown in Fig. 2. The capacity of an edge e is denoted by c_e, its routing demand by u_e. Let $o_e = \max(u_e - c_e, 0)$ be the overflow of an edge e. The total overflow of the layout is given by $\sum_{\text{all } e} o_e$, and the maximal overflow given by $\max_{\text{all } e} o_e$.

In relation with the dummy fill metric, D, we define the *initial pattern density* (IPD) in a region as the ratio of the area of metal in the region to its total area. In our routing model, we assume reserved horizontal/vertical routing layers, and separately consider routing edges in the horizontal and vertical directions. In practice, since our algorithm will associate the increase in the EPD with edges of the GRG, we use a shifted grid for CMP computations. On the horizontal layer, each CMP grid/tile around an edge e has the same size as a GRG grid, but is centered about the edge instead of the vertex (i.e., it is offset to the right by half a grid relative to the GRG grid centered at the left endpoint of e). A CMP tile on the vertical layer is similarly defined.

Next, we introduce the oxide CMP model in [14]. In this model, the ILD thickness z at location (x, y) in the layout can be calculated using the following formula:

$$z = \begin{cases} z_0 - [K\tau/\rho(x,y)] & \tau \leq (\rho z_1/K) \\ z_0 - z_1 - K\tau + \rho(x,y)z_1 & \tau \geq (\rho z_1/K) \end{cases}, \quad (1)$$

where K is the blanket oxide polishing rate, z_0 is the thickness of oxide deposition, z_1 is the initial step height, τ is the total polish time, and $\rho(x, y)$ is the EPD in location (x, y) before oxide CMP. A schematic that describes the variables can be found in the Fig. 3 in [14]. The variables K, z_0, z_1 and τ are constants for a specific CMP process.

Generally, the total polish time τ is larger than $(\rho z_1/K)$, and therefore, the final oxide thickness z is between 0 and $(z_0 - z_1)$. As a consequence, the final ILD thickness in different locations has an affine relationship with the EPD in that location. The *effective pattern density* (EPD) can be calculated by convolving the IPD with the weighting function [14]:

$$f(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right). \quad (2)$$

Here, $f(x, y)$ is a Gaussian function with standard deviation σ. This function can be discretized to the tile grid, and truncated beyond a *weighting window* of size $L \times L$ distance units, equivalent to $(2l + 1) \times (2l + 1)$ tiles. For this Gaussian function, $\sigma = L/2$, equivalent to $\lfloor (2l + 1)/2 \rfloor$ tiles, and the value of $f(x, y)$ will be truncated to 0 beyond the weighting window. If we discretize the value of the weight function in tile t_{ij} to $f(i, j)$, and denote the IPD of t_{ij} as d_{ij}, then the EPD of t_{ij}, ρ_{ij}, can be calculated by a circular convolution as [16]:

$$\rho_{ij} = \sum_{n_1=-l}^{l} \sum_{n_2=-l}^{l} d_{i+n_1, j+n_2} \times f(n_1, n_2). \quad (3)$$

Thus, the EPD of a tile is calculated as a weighted sum of IPDs.

Dummy fill optimization inserts the minimum fill, D, to ensure a spatially even EPD, i.e.,

$$\text{ran}(\rho) = \rho_H - \rho_L \leq \epsilon, \quad (4)$$

where ϵ is a user-specified parameter, and $\text{ran}(.)$ is an operator that finds the range of a function, i.e., the difference between its maximum and minimum. Here, ρ_H (ρ_L) is the maximum (minimum) value of the EPD, ρ, over the layout.

Methods for dummy fill insertion can be classified into two categories: rule-based and model-based. Rule-based methods reduce the CMP variation by inserting dummy fill to ensure that the IPD in each region meets a certain threshold. While these methods are simple to execute, they are heuristic in nature and do not guarantee optimality. Model-based methods use CMP models such as the one introduced above, and directly optimize the variation of the EPD globally based on the model. These methods are more computational, but can use a significantly lower amount of dummy fill [16]. The work in [16] presents a linear programming based algorithm to optimize the total amount of dummy fill, D, with constraints on $\text{ran}(\rho)$. In this paper, we use D as the metric to evaluate the different global routing solutions.

3. PREVIOUS WORK

Our routing algorithm is based on NTHU-Route 2.0 (NTHR) [1], which is briefly reviewed in this section. There are four stages in the NTHR algorithm: the initial stage, the main stage, the refinement stage, and the layer assignment stage.

The purpose of the initial stage is to generate an initial global routing solution. First, a multi-layer design is projected to a 2D plane, and then FLUTE [4] is used to decompose each multi-pin net into a set of two-pin nets. Next, NTHR sets up the probabilistic congestion map by adding half a unit of demand to each edge on the two probabilistic L-shape routes, or a full demand to each edge on a straight route. Then the topology of every multi-pin net is modified using the edge-shifting technique [15]. Finally, every two-pin net is routed by L-shaped pattern routing. During this stage, the cost function in [15] is used to calculate the cost of an edge:

$$\mathcal{C}_e' = 1 + \frac{p_3}{1 + e^{-p_4(u_e - c_e)}}, \quad (5)$$

where p_3 and p_4 are user-defined parameters. In NTHR, $p_3 = 0.8$, $p_4 = 2$.

In the main stage, the initial solution is improved by iteratively ripping up and rerouting (RRR) every congested two-pin net. A

two-pin net is considered to be congested if there are one or more overflowing edges on its path. NTHR uses a technique of identifying congested regions to choose the ordering for RRR. In the RRR process, each ripped-up two-pin net is first rerouted by monotonic routing [15], and then by the adaptive multi-source multi-sink maze routing method, if an overflow-free path cannot be found by monotonic routing. The RRR process is repeated until the total overflow is no more than a predefined threshold or the number of iterations reaches a predefined value. A history-based cost function is used in this stage, and the basic form is as follows:

$$\mathcal{C}_e^{''} = X_e \times B_e + H_e \times G_e + V_e \times B_e, \tag{6}$$

where $\mathcal{C}_e^{''}$ is the cost of edge e; X_e is the wire length cost, which is set to 1 since the wire length will increase by 1 when routing through e; G_e is congestion cost, H_e is the cost for historic overflow; V_e is via cost; B_e is a factor defined as follows:

$$B_e = 1 - e^{-\alpha e^{-\beta i}}, \tag{7}$$

where α and β are user-defined parameters, and i is the current iteration count. In NTHR, $\alpha = 5$, and $\beta = 0.1$. Thus, B_e will be bounded between 1 and 0, and will decrease as i increases. By incorporating B_e into the wire length cost and via cost, the congestion cost will gradually take the dominant role in the total cost as the iteration number increases, which is helpful for NTHR to obtain paths without overflow.

The main purpose of the refinement stage is to search an overflow-free path for every congested two-pin net, which is adapted from the main stage with the following two major differences. First, while in the main stage, the congested region identification technique is used to determine the RRR order, in the refinement stage, NTHR rips up and reroutes every congested two-pin net in the non-increasing order of the number of overflowed edges on its path of the net. Second, the cost of an edge e is defined as follows:

$$\mathcal{C}_e^{'''} = \begin{cases} 1 & \text{if edge } e \text{ has overflow,} \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

For multi-layer designs, the layer assignment stage is performed to map the routing solution from the projected 2D plane to the original multiple layers with the algorithm in [11].

4. THE FLOW OF THE NEW ROUTING ALGORITHM

There are two major differences between the proposed routing algorithm and the original NTHR.

Firstly, we augment NTHR so that the router becomes CMP-aware. We study several possible metrics related to dummy fill and their correlations with the amount of inserted dummy fill, D, and then elaborate effective cost functions to be integrated into NTHR to perform dummy fill optimization. A detailed presentation of the cost functions used to guide dummy fill optimization is provided in Section 5.

Secondly, we add another stage after the refinement stage in order to reduce D further by ripping up and rerouting the nets passing through the tiles related to ρ_H[1]. This stage, which we call *the EPD postprocessing stage*, has the following steps. We initially identify the tile t_k that has the maximum EPD in the layout, and then find all the 2-pin nets whose paths pass through the edges in the weighting window of t_k. Note that the maximal EPD is attributed not only to wires in t_k but also to wires in other tiles within the weighting window of t_k. Next, we sort all of these 2-pin nets in nonincreasing order of the minimal distance of the two pins to t_k. The motivation is that for the 2-pin nets further from t_k, and it should be easier to find an alternative path that does not pass through the weighting window of t_k. In case of ties, we employ other sorting criteria, such as the number of edges that are in the weighting window of t_k and also on the path of a 2-pin net, and the total overflow of the path of a 2-pin net.

After this step, we rip up and reroute the nets one by one using a method adapted from the refinement stage of NTHR. The goal is to find a path to decrease ρ_H, and at the same time ensure that the overflow is not increased. The cost function used here will be introduced in Section 5.3. In the process of ripping-up and rerouting, once ρ_H is decreased sufficiently, and the tile with new maximum EPD is different from t_k, a new iteration will start; otherwise, if ρ_H can not be decreased after trying all the 2-pin nets identified in an iteration, this stage stops. In practice, to control the runtime, we monitor the improvement in ρ_H, and then if its percentage reduction over the past N_{stop} RRR iterations is smaller than ϵ_{stop}, we stop this stage. We calibrate reasonable values of these parameters as $N_{stop} = 100$ and $\epsilon_{stop} = 1\%$. In an iteration, after rerouting a net, if the maximal EPD or total overflow is larger than that before ripping up it, the original path is restored.

In rare instances, there may be more than one tiles with EPD equal to ρ_H. In this case, we just first process the tile found first. Then after a couple of rip-up-and-reroute operations, it is likely that the EPD of this tile becomes smaller than ρ_H, and next we process another tile with EPD equal to ρ_H. In other words, the tie is broken arbitrarily, and then the other tile(s) with the same EPD will be considered soon in a subsequent iteration.

5. COST FUNCTION

The success of the routing framework is critically dependent on the choice of a cost function. It is vitally important for the cost function to be computationally easy to evaluate, and yet hold enough fidelity to capture a more complex underlying objective that it represents. Therefore, we elaborate efficient and effective cost functions to achieve our goal.

To address the new objective of minimizing D, we define Φ_e as the dummy fill cost of edge e, which will be integrated to the original cost function used in NTHR. The remainder of this section will first discuss the computation of Φ_e, and then describe how it is used at various stages of the routing process.

5.1 Finding a surrogate for the dummy fill cost

Our cost function requires the computation of Φ_e, which is related to the dummy fill metric, D. The direct calculation of D is highly computational [16], and therefore, we have to find a metric which correlates well with D and is easy to use in the routing process.

As a first step, we present a metric Γ that is linearly related to D:

$$\Gamma = \sum_{t_{ij} \in Q_1} (\rho_U - \rho_{ij}), \tag{9}$$

where $\rho_U = \rho_H - \epsilon$, which is the lower bound on the EPD after a dummy fill procedure achieves the constraint described in (4), and Q_1 is the set of tiles for which $\rho_{ij} < \rho_U$ before inserting dummy fill. For these tiles, ρ_{ij} must be increased by inserting dummy fill. Note that unlike $\mathrm{ran}(\rho)$, which merely captures the difference between the maximum and the minimum values of ρ, but is insensitive to the distribution of ρ within this range, this metric captures the distribution of ρ over all the related tiles.

We will now present results that link Γ to D.

Lemma 1 *Given a weighting function $f(i,j)$ whose value is truncated to 0 outside the weighting window,*

$$\sum_{\text{all } t_{ij}} \rho_{ij} = b \sum_{\text{all } t_{ij}} d_{ij}, \text{where } b = \sum_{i=-l}^{l} \sum_{j=-l}^{l} f(i,j).$$

The proof of Lemma 1 is provided in the Appendix. The above lemma may be used to show a key result that drives our approach.

[1] In other stages, we do not adopt this measure to reduce ρ_H, but integrate the dummy fill cost function into the router to minimize ρ_H, because routability has the highest priority at those stages.

Theorem 1 *Let Q_0 be the set of the tiles with EPD no less than ρ_U before dummy filling, and Q_1 be the complement of Q_0. Let d'_{ij} (ρ'_{ij}) be the IPD (EPD) of t_{ij} after dummy filling, and let*

$$\mu = \sum_{t_{ij} \in Q_0} (\rho'_{ij} - \rho_{ij})$$

$$\nu = \sum_{t_{ij} \in Q_1} (\rho'_{ij} - \rho_U)$$

Then

$$D = c\,(\Gamma + \mu + \nu)\,, \tag{10}$$

where $c = 1/b > 0$, and b is defined in Lemma 1.

Again, for the proof, the reader is referred to the Appendix.

Note that only the tiles in set Q_1 require the insertion of dummy fill. In practice, since the minimal amount of dummy fill is inserted, it can be expected that after dummy fill insertion, ρ'_{ij} of $t_{ij} \in Q_0$ including the tile with ρ_H will remain unchanged or, at worst, increase by a very small amount. Therefore, μ is a small number compared to Γ. Moreover, ρ'_{ij} for $t_{ij} \in Q_1$ will be approximately equal to or just a little larger than ρ_U, and therefore ν is also a small number compared to Γ. In the following analysis, we will always assume that μ and ν are negligible compared to Γ. These *dummy fill assumptions* imply that

$$D \approx c\Gamma. \tag{11}$$

This relationship will be demonstrated experimentally in Section 6. Since $c = 1/b$, and is a positive constant for a given weighting function, to minimize D, we should minimize Γ.

Though Γ is much easier to compute than D, it is still too complex to be used directly in routing process, since it requires an enumeration over all tiles in Q_1; as we will see, the cardinality of this set can be very large. In order to find another simpler metric which can be used in routing, next we will analyze the impact of our routing procedure on Γ, and then D.

In the routing process, every two-pin net passing through overflowing edges or passing through the weighting windows of the tile(s) with ρ_H is ripped up and then rerouted one by one in some ordering. Next we will present two theorems to reveal the impacts on Γ of ripping up and rerouting a single two-pin net: the first theorem is for rerouting a two-pin net, and the second one for ripping up a two-pin net.

Theorem 2 *Consider a partial routing solution $\mathcal{S}$, and the solution $\bar{\mathcal{S}}$ after routing one more net, and consider the contribution of each edge e of this net. Let $\Delta\Gamma$ and $\Delta\rho_H$ be, respectively, the change in Γ and ρ_H from $\mathcal{S}$ to $\bar{\mathcal{S}}$ due to edge e.*
(a) The following inequality holds:

$$\Delta\Gamma \geq |Q_1| \cdot \Delta\rho_H - b \cdot \Delta_d^M, \tag{12}$$

where Δ_d^M is the increase in the IPD of a tile when routing through one edge in the tile, and is equal to the ratio of the area occupied by a wire track to that of a tile.
(b) If $\Delta\rho_H = 0$, $\bar{Q}_1 = Q_1$ and $W_e \subseteq Q_1$, then

$$\Delta\Gamma = -b \cdot \Delta_d^M, \tag{13}$$

where $\bar{Q}_1$ is the new Q_1 after routing through e, and W_e is the weighting window of the tile associated with e.

The proof is detailed in the Appendix. It is important to point out that (12) holds regardless of whether $\Delta\rho_H > 0$ or $\Delta\rho_H = 0$.

Corollary 1 *Consider a partial routing solution $\bar{\mathcal{S}}$, and the solution $\mathcal{S}$ after ripping up one more net, and consider the contribution of each edge e of this net. Let $\Delta\Gamma$ and $\Delta\rho_H$ be, respectively, the change in Γ and ρ_H from $\bar{\mathcal{S}}$ to $\mathcal{S}$ due to edge e. Then*

$$\Delta\Gamma \leq |Q_1| \cdot \Delta\rho_H + b \cdot \Delta_d^M, \tag{14}$$

where Δ_d^M has the same meaning as in Theorem 2.

The proof is simple. Ripping up a net N from $\bar{\mathcal{S}}$ is a symmetric process of starting from S and routing net N with the same routing path. In this case, the change in Γ (ρ_H) from S to $\bar{\mathcal{S}}$ is $-\Delta\Gamma$ ($-\Delta\rho_H$), using the terminology defined in the statement of Theorem 2. Therefore, $-\Delta\Gamma \geq |Q_1| \cdot (-\Delta\rho_H) - b \cdot \Delta_d^M$. This corollary holds when $\Delta\rho_H \leq 0$. □

Table 1: The cardinality of set Q_1 for ISPD07 circuits

Circuits	Horizontal layer		Vertical layer	
	#tiles	$\lvert Q_1\rvert$	#tiles	$\lvert Q_1\rvert$
adaptec1	6561	4566	6561	4542
adaptec2	11236	10557	11236	10435
adaptec3	24180	21356	24180	20874
adaptec4	24180	22925	24180	22469
adaptec5	24180	18119	24180	19181
newblue1	6400	5531	6400	5766
newblue2	28830	24619	28830	23964
Average	1.00	0.85	1.00	0.85

In practice, $|Q_1|$, the cardinality of set Q_1, is seen to be very large. Table 1 shows the values of $|Q_1|$ for the routing solutions obtained by NTHR for the 2D ISPD07 benchmarks[2]. We can see that on average 85% of the tiles are in Q_1. The above theorems imply that if the cardinality of set Q_1 is large, then $\Delta\Gamma$ will be significant even for a small change in ρ_H, which implies a nontrivial change in D. In other words, D is sensitive to the change in ρ_H due to the large cardinality of set Q_1.

Example: For the benchmark newblue2, $b = 0.532$, $\Delta_d^M = 7.41 \times 10^{-4}$. We use $|Q_1| = 23964$, the value for the vertical layer. Suppose routing through edge e causes a small increase[3] of ρ_H, and $\Delta\rho_H = 2.36 \times 10^{-6}$. By computing the lower bound for $\Delta\Gamma$ using (12), and using this to predict a lower bound on ΔD using (11), it can be shown that the increase in D is equivalent to at least 142 tracks. By Corollary 1, if $\Delta\rho_H$ is reduced by the same amount in the above example, D will reduce by at least 142 tracks. This example shows clearly that even though ρ_H changes by a very small amount, D will change greatly due to the large cardinality of Q_1. □

The analysis above shows that D is highly sensitive to the change in ρ_H. In other words, minimizing ρ_H is a good surrogate for minimizing D. Therefore, our routing objective is to minimize the ρ_H after routing finishes. We do this by trying to leave ρ_H unchanged, or minimizing the increase in its value, when routing each net, and by trying to reduce ρ_H when ripping up a net in the EPD postprocessing stage. Intuitively, this objective tries to ensure that all ρ values are low and as well-balanced as possible.

5.2 Dummy fill cost function

In NTHR, monotonic routing and maze routing are used in the RRR process. When rerouting a net, the costs of edges in the searching region are used to guide the router to find a new path for the net. In order to optimize D, we compute the dummy fill cost for every edge e, denoted as Φ_e, and then integrate Φ_e to the router.

Based on the results in Section 5.1, we now show how we compute the dummy fill cost, Φ_e. Our approach is based on the previous analysis, which shows that minimizing ρ_H in routing process is a good surrogate for minimizing D. To capture this objective well, the following three aspects should be considered in the dummy fill cost function.

One possible component of the cost function could be to determine the effect of routing through an edge e on the increase in ρ_H. In this case, a large cost should be assigned to e as a penalty. To

[2] The characteristics of benchmarks used are listed in Table 2.

[3] As a reference, the values of ρ_H for the circuits in our benchmarks are between 0.10 and 0.20 for both the horizontal and the vertical layers.

achieve this, the following term could be used:

$$\Omega_e = \begin{cases} \exp\left(\frac{\Delta\rho_H}{\Delta_\rho^M}\right) & \text{if } \Delta\rho_H > 0, \\ 0 & \text{otherwise,} \end{cases} \quad (15)$$

where $\Delta\rho_H$ is the change in ρ_H after routing through edge e, and Δ_ρ^M is the possible maximal increase in the EPD of any tile and equal to the increase in the EPD of the tile associated with edge e, when a wire is routed through edge e. Here the role of Δ_ρ^M is to normalize the numerator to a value between 0 and 1. Since Ω_e captures the direct increase of ρ_H, the exponential function is used to magnify the penalty.

However, as we will soon see, such a function is not general enough since it only considers the contribution of a single edge, rather than that of a path. In particular, it is possible that the cost of each single edge e_i on path P is 0 by (15), since no single edge increases the value of ρ_H; however, routing through path P may still increase ρ_H due to the cumulative effects of all edges on the path, due to which the dummy fill cost of path P should not be 0.

For example, consider a tile t_k with large EPD which is near a path P, and let W_k be the weighting window of tile t_k, as shown in Fig. 3. In this example, due to congestion, the net $S \rightarrow T$ is detoured and path P shown in the figure is chosen as a candidate. The edges of P that lie along wire segments AB and CD are within W_k, and together, these may increase ρ_k by a significant amount. Before routing this net, if ρ_k was smaller than $\rho_H - \Delta_\rho^M$ but still close to ρ_H, then after routing through P, it is likely that ρ_k may exceed ρ_H. However, Eq. (15) may not capture this effect, since each edge individually may have zero cost according to this function, and therefore we have to build a new cost component to address this problem.

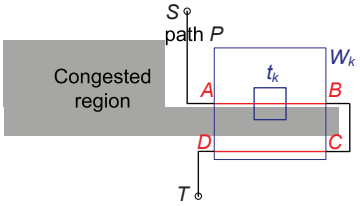


Figure 3: Routing through path P may increase the EPD of tile t_k significantly.

Let S_P denote the set of all paths that may be used to route the net that is currently under consideration. Let J_P be the increase in the EPD of a tile t_k, if route P is chosen, and $J_M = \max_{S_P} J_P$. Then for tile t_k, if $\rho_k < \rho_H - J_M$, then ρ_k cannot exceed ρ_H after routing through any path; otherwise, ρ_k may become larger than ρ_H, depending on which path P is chosen. To deter the router from choosing a path that creates this violation, we assign a large penalty to the edges within W_k to deter the router from choosing a path that contains edges in W_k. From the point of view of the cost associated with an edge e, if there is a tile t_k within W_e with $\rho_k \geq \rho_H - J_M$, a large penalty should be assigned to e and the penalty should increase for higher ρ_k, where W_e is the weighting window of the tile associated with e. We define

$$\Delta\rho'_H = \max_{t_k \in W_e} (\rho_k + J_M - \rho_H), \quad (16)$$

where the role of the max function is to determine the largest possible increase in ρ_H. Then we use the following as the first component of the cost function for e:

$$\Theta_e = \begin{cases} \exp\left(\frac{\Delta\rho_k}{\Delta_\rho^M}\right) \cdot \exp\left(p_0 \cdot \frac{\Delta\rho'_H}{\Delta_\rho^M}\right) & \text{if } \Delta\rho'_H > 0, \\ 0 & \text{otherwise,} \end{cases} \quad (17)$$

where $\Delta\rho_k$ is the increase of ρ_k after routing through edge e, and $\Delta\rho'_H$ is the adapted version of $\Delta\rho_H$, and p_0 is a user-defined parameter to control Θ_e in a proper range. In our experiments, p_0 is tuned on circuit newblue2 to make $\max(\Theta_e) = 16.3$. Here, the first exponential term is used to capture the fact that the larger $\Delta\rho_k$ is, the larger is the possibility that ρ_k will exceed ρ_H. The term Δ_ρ^M is used to normalize the numerators.

The computation of J_M above requires the determination of a realistic upper bound of the increase in the EPD of tile t_k over all routes for the net being considered. The most pessimistic estimate assumes that when path P passes through all the edges in W_k, J_P reaches its maximum. However, this case is excessively pessimistic and extremely rare even in a very congested design, since practical routes do not go through so many bends and detours within a small region. A reasonable practical upper bound for J_M corresponds to the case shown in Fig. 3, where a "U"-shaped path is used to detour and passes through the weighting window of t_k twice. In this case, J_M can be calculated as:

$$J_M = \Delta_d^M \sum_{i=-l}^{l} \sum_{j=0}^{1} f(i,j), \quad (18)$$

where Δ_d^M is the same as in Theorem 2, l the same as in (3), and $f(i,j)$ is the weighting function. To be safer still, our implementation uses twice the calculated value of J_M in (18) as the guardband.

A second component of the cost function can be determined as follows. For the tiles whose EPDs are close to ρ_H, routing through the edges in these tiles will increase their EPDs further and it is likely that their EPDs will exceed the current ρ_H soon in the routing process. To control this trend, we assign large cost penalties to the edges associated with such tiles. On the other hand, we do not want the CMP optimization to affect the routing with normal objectives such as wire length and overflow too much, and therefore, for the tiles with EPD values not large enough to cause the increase of ρ_H, their costs should be small. To achieve this goal, we adapt the cost function (5) in [15] to use in our work, since it increases very slow when the variable is not close to a limit. We remove the constant factor 1 in the cost function to make its low bound to be 0, and normalize the EPD value of a tile by the current ρ_H. Then the cost of an edge e is:

$$\Psi_e = \frac{p_1}{1 + e^{p_2(1-\rho_e/\rho_H)}}, \quad (19)$$

where ρ_e is the EPD of the tile associated with edge e, p_1 and p_2 are user-defined parameters. In this work, we choose $p_1 = 4$ and $p_2 = 10.99$, which makes $\Psi_e = 1$ when $\rho_e = 0.9 \times \rho_H$, the cost equal to that of wire length, and makes $\Psi_e = 2$ when $\rho_e = \rho_H$. Here, $\Psi_e = 2$ is the maximum of Ψ_e, which is double of the cost of wire length. On the other hand, when $\rho_e = 0.7 \times \rho_H$, $\Psi_e = 0.14$, which is rather small and will not affect the routing too much.

The total dummy fill cost of edge e is defined as follows:

$$\Phi_e = \Psi_e + \Theta_e \quad (20)$$

Here, Ω_e is not used since it is covered by Θ_e. Also note that the parameter p_0 in (17) will modulate Θ_e to adjust the ratio of Θ_e to Ψ_e and to determine the maximal value of Φ_e. Though our cost function is essentially heuristic, we expect it to work well due to the following two factors. First, it considers not only the impact of routing through one tile/edge on its own EPD, but also the impact on the neighboring tiles, which captures the long range effect of CMP variation. Second, our cost function is consistent with the previous theoretical judgment: it always tries to control and avoid the increase of ρ_H, which is desirable for the minimization of D.

5.3 Cost function in different stages

In this section, we will introduce how the proposed cost functions are integrated into the routing framework.

First, we point out that we do not integrate the dummy fill cost functions to every stage of NTHR. In NTHR, the main purpose of the initial stage is to obtain the initial congestion map and the initial routing solution for later use, and the space for optimizing all the objectives together is limited since L-pattern routing is used. Furthermore, from our empirical observation, the addition of dummy

fill optimization in the initial stage does not improve the final solution but increases the runtime of later stages. The principle here is that the initial stage is very useful in controlling conventional routing metrics, and dummy fill can be effectively optimized in the later stages of NTHR. Therefore, we do not add a dummy fill optimization objective in the initial stage. We also do not perform dummy fill optimization in the layer assignment stage in our current work, since we use the single-layer CMP model in this work as a first step to dummy fill optimization, and leave multi-layer dummy fill optimization as future works.

In the main stage, we consider the dummy fill optimization by integrating the proposed cost function with traditional cost function of NTHR. In the main stage, the cost of edge e becomes as follows:

$$\mathcal{C}_e = \mathcal{C}_e^{''} + \gamma_1 \times \Phi_e \times \max(B_e, 0.1), \tag{21}$$

where $\mathcal{C}_e^{''}$ is the original cost function (6) in NTHR, Φ_e is the dummy fill cost function (20), B_e is a scaling factor defined in (7) and γ_1 is the user-defined weight for Φ_e. In our implementation, γ_1 is tuned on circuit newblue2 to be 2.5. The mechanism of using term $\max(B_e, 0.1)$ is similar to that of using B_e in the original cost function (6) in NTHR: when the number of RRR iterations increases, which means it is more and more difficult to route the current net, B_e will decrease from 1 gradually towards 0, and then dwarf all the cost function components but the congestion cost, which helps the router to obtain a path without overflow. However, to prevent ρ_H from increasing by a large amount when B_e becomes too small, we use 0.1 as the lower bound of dwarfing dummy fill cost. From our empirical observation, the value of 0.1 achieves a good balance between routability and dummy fill optimization.

In the refinement stage, most nets have been routed in the previous stages, and only a few nets must be ripped up and rerouted due to overflow. Since the primary goal is to reduce the overflow with limited routing resources, high priority should be given to reduction of overflow, and thus the following cost function is used:

$$\mathcal{C}_e = \gamma_2 \times \mathcal{C}_e^{'''} + \Phi_e, \tag{22}$$

where $\mathcal{C}_e^{'''}$ is the original cost (8) used in NTHR, Φ_e is the dummy fill cost, and γ_2 is an empirically chosen weight whose value should be large enough to make $\mathcal{C}_e^{'''}$ dominant to give a high priority to the reduction of overflow. We choose $\gamma_2 = M_P \cdot \max(\Phi_e)$, where M_P is an upper bound on the length of the longest possible path for a 2-pin net in the layout. In this way, a path with smallest overflow will always be chosen by the router. When there are several candidate paths with zero overflow, the dummy fill cost will guide the router to choose the best one out of them. In our implementation, we choose $M_P = N_g$ in choosing γ_2, where N_g is the total number of grids in the layout: this is a realistic estimate of the upper bound. In estimating $\max(\Phi_e)$, we note that $\max(\Psi_e) = 2$ (from (19)), and that the value of $\max(\Theta_e) = 16.3$; this is based on empirical tuning on circuit newblue2, as will be explained in Section 6. Therefore, we set $\max(\Phi_e) = 18.3$.

In the EPD postprocessing stage that we introduce, our goal is to reduce the maximal EPD further but not to increase overflow, and therefore the cost function used is the same as (22).

5.4 Why not care about ρ_L

As seen in (4), the range of the EPD (and hence the range of CMP variation) can be reduced by either reducing ρ_H or by increasing ρ_L. However, our arguments above primarily focus on reducing ρ_H in order to reduce D. A natural question to ask is whether it would also be useful to make efforts to increase the value of ρ_L in order to reduce D. Here, "make efforts" means taking measures similar to what we have done for minimizing ρ_H, e.g., a bonus is given to an edge or a path when routing through it will increase ρ_L.

Given a layout, let us consider the change of D after routing through an edge e. We consider two cases for routing e:

- If $\Delta\rho_H > 0$, then as shown by the argument and example after Theorem 2, D could increase by a large amount due to the large cardinality of set Q_1, so we elaborate cost functions to avoid using edge e on the routing path.
- In contrast, if $\Delta\rho_H = 0$, then as revealed by Theorem 2, $\Delta\Gamma \geq -b\Delta_d^M$, i.e., the largest possible reduction in Γ is $b\Delta_d^M$. Furthermore, by Theorem 2, if the three conditions $\Delta\rho_H = 0$, $\bar{Q}_1 = Q_1$ and $W_e \subseteq Q_1$ are satisfied, then $\Delta\Gamma = -b\Delta_d^M$. In other words, routing through any edge which satisfies these three conditions will achieve the *same* maximum reduction $b\Delta_d^M$ on Γ, *no matter how* ρ_L *changes*. Therefore, there is no reason we should make extra efforts to route through a few special edges, that are in the weighting window of the tile(s) with EPD equal to ρ_L, to increase ρ_L. By (11), the above analysis also holds for D, and we do not need to care about how ρ_L changes in terms of minimization of D.

The argument for routing through a path is similar to the analysis above: in order to obtain the same change in D, we do not need to make extra efforts to let the path pass through a few special edges to increase ρ_L, because there are many other choices of edges to pass through, which have the same effect on D. Based on the analysis above, in our algorithm, we focus our efforts to minimize ρ_H but do not make efforts to increase ρ_L.

6. EXPERIMENTAL RESULTS

We have implemented the algorithm in C++, and have tested it on a 64-bit Linux machine with an Intel® Core(TM)2 Duo 3.00GHz CPU and 8GB memory. The routing algorithm is implemented as a program with two options: MaxEPD optimizes the dummy fill based on minimizing the maximum EPD, and NoCMP does not consider dummy fill.

We have tested our routing scheme on the 2D ISPD07 benchmarks [6, 13], whose characteristics are listed in Table 2. In the table, the column "guardband" lists the percentage of the given capacity to the actual capacity after the guardband adjustment. The unit for grid size is the number of wire tracks. Since newblue3 is unroutable using current routers [7], it is not considered in our experiments. We assume 0.13um technology is used for the circuits, and the wire width equal to 5 times of the minimal value, 0.65um. We use a typical planarization length, L = 1mm. The size of a CMP tile (which is used to evaluate IPD and EPD) is set to about 100um to obtain a good balance between accuracy and performance[4]. As a result, there will be about 121 CMP tiles in the weighting window of a CMP tile.

Table 2: Benchmark information

Circuits	Grid dimension	Grid size	Guard-band	#Nets	Grid HPWL (×1.e5)
adaptec1	324 x 324	35	90	176k	30.00
adaptec2	424 x 424	35	100	208k	28.82
adaptec3	774 x 779	30	90	368k	86.20
adaptec4	774 x 779	30	90	401k	81.75
adaptec5	465 x 468	50	100	548k	88.97
newblue1	399 x 399	30	90	271k	20.80
newblue2	557 x 463	50	100	374k	41.91

To demonstrate the effectiveness of our cost functions, we compare MaxEPD with the method proposed in [17], which attempts to minimize the CMP variation using maze routing under a CMP-aware cost function, in which the cost of an edge is the EPD of the associated CMP tile. Since the codes in [17] are not available for public access, in order to compare with this method, we developed another router by replacing the cost function we propose in MaxEPD by the cost function in [17][5]. We denote this router as "Yet

[4] In this case, a CMP tile used to compute IPD and EPD may contain more than one routing grids. This explains why the number of tiles shown in Table 1 is much smaller than that of grids shown in Table 2.

[5] The cost function used in the global routing stage in [8] is in fact the same as that used in [17].

Table 3: Comparison of routing results among NoCMP, YaCMP and MaxEPD

Circuits	Total overflow			Wire length (×1.e5)			$\mathrm{ran}(z)$ (Å)			Runtime (s)		
	NoCMP	YaCMP	MaxEPD	NoCMP	YaCMP	MaxEPD	NoCMP	YaCMP	MaxEPD	NoCMP	YaCMP	MaxEPD
adaptec1	0	0	0	42.51	44.33	45.41	2145	2038	1817	285	998	2784
adaptec2	0	0	0	40.01	40.48	41.19	2622	2557	2390	61	113	265
adaptec3	0	0	0	108.95	112.97	114.08	2251	2169	1875	305	1080	1229
adaptec4	0	0	0	102.34	102.76	103.13	2138	1950	1806	71	132	195
adaptec5	0	0	0	121.13	122.25	127.15	2628	2582	2522	668	1215	3371
newblue1	3	16	6	32.42	32.65	32.87	2337	2262	2230	227	683	652
newblue2	0	0	0	57.13	57.59	57.89	1839	1717	1603	35	89	132
Summary	3	16	6	1.000	1.017	1.033	1.000	0.955	0.889	1.000	2.592	4.658

another CMP-aware router" (YaCMP).

For MaxEPD, we tune the weights p_0 in (17) and γ_1 in (21) with circuit newblue2[6], and then use these weights for all the circuits. Specifically, we use $\gamma_1 = 2.5$, and p_0 is tuned so that $\max(\Theta_e) = 16.3$. These weights determine the tradeoff between traditional routing and dummy fill minimization, and therefore, they should be in an appropriate range. For completeness, we will list newblue2 in our tables of results, but the gains on this circuit can be appropriately discounted since the parameters were tuned on it. For YaCMP, we also tune the weights γ_1 in (21) for the circuit newblue2 in the similar way. Specifically, we use $\gamma_1 = 5.5$.

Table 3 compares the routing results from NoCMP, YaCMP and MaxEPD. The last line, labeled "Summary," presents a synopsis of the comparison between the three methods. For each circuit, we show the total overflow, wire length, as well as the unevenness in the topography after CMP. In the table, "$\mathrm{ran}(z)$" stands for the range/variation of the ILD thickness. The value of $\mathrm{ran}(z) = z_1 \cdot \mathrm{ran}(\rho)$ is calculated according to (1), suppose z_1 = 7000 Å [16]. Note that each via is counted as 1 unit of wire length in the calculation of the total wire length, according to the rule in ISPD 2008 global routing contest [7]. For all the circuits except newblue1, the maximal overflow is 0. For newblue1, the maximal overflows for the routing solutions obtained by NoCMP, YaCMP and MaxEPD are all equal to 1. Since the guardband factor for newblue1 is 90%, the one or two overflows on an edge can be eliminated in later stage with the reserved capacity, and then these overflows will not cause any problems to the EPD computation and dummy filling.

It can be seen that MaxEPD consistently provides significant improvements in the ILD variations, at the cost of a small increase in the wire length and overflow (only for the circuit newblue1). Compared with NoCMP, MaxEPD improves the post-CMP ILD variations by 11.1% on average and up to 16.7%; compared with YaCMP, the improvement is 7.0% on average and up to 13.5%.

The runtime of MaxEPD is acceptable, even for the large circuit adaptec5. The runtime for circuit adaptec1 is much longer than NoCMP, compared with the increase in runtime for other circuits. This is likely because the weights which are tuned for circuit newblue2 are not appropriate for circuit adaptec1. Note that it is difficult to route circuit newblue1 without overflow [7], and therefore there is small space to optimize CMP variation. As a result, the improvement in the CMP variation, for both MaxEPD and YaCMP, is small.

As a verification step, Table 4 shows an evaluation of the quality of our results, using the ranged-variation linear programming (LP) formulation of the dummy fill algorithm in [16]. We set $\epsilon = 0.02$ in (4). A commercial LP solver, ILOG CPLEX® [5], is used to solve the dummy filling problem on a 64-bit Linux machine with a 2.6 GHz AMD® Opteron® 2218 processor and 2GB memory. In the table, "fillWL/minWL" presents the ratio of the equivalent wire length of total dummy fill to the minimal total wire length (minWL). The value of minWL is the sum of the minimal wire length without detours, ignoring congestion constraints. Compared with NoCMP, the MaxEPD approach significantly reduces the total fill by 22.0% on average and up to 41.5%; compared with YaCMP, MaxEPD reduces the fill amount by 14.1% on average and up to 23.6%. The data show the effectiveness of our routing algorithm, especially the proposed cost functions and the strategy of minimizing the maximal EPD. The last column of the table shows the runtime of the dummy fill algorithm of [16] for MaxEPD. The CPU time required by the dummy filling step, applied to the results of NoCMP and YaCMP, is similar and on average within 2% of that of MaxEPD.

[6]Circuit newblue2 is chosen to tune the weights due to its medium size and least runtime among all the circuits. Generally, tuning weights costs tens of times the runtime of a single run. Also note that p_1, p_2 and γ_2 take the pre-defined values and are not required to be tuned.

Table 4: Comparison of dummy fill results among NoCMP, YaCMP and MaxEPD. The data in column 2 and 3 are normalized with the basis case (1.0) corresponding to NoCMP.

Circuits	fillWL/minWL		Fill time (s)
	YaCMP	MaxEPD	MaxEPD
adaptec1	0.831	0.585	224
adaptec2	0.956	0.852	495
adaptec3	0.917	0.700	1849
adaptec4	0.849	0.736	2083
adaptec5	0.950	0.873	1848
newblue1	0.944	0.920	206
newblue2	0.890	0.794	2775
Summary	0.905	0.780	

Table 5: Comparison of IPD gradient $\mathcal{G}$ among NoCMP, YaCMP and MaxEPD

Circuits	$\mathcal{G}$ for horizontal layer			$\mathcal{G}$ for vertical layer		
	NoCMP	YaCMP	MaxEPD	NoCMP	YaCMP	MaxEPD
adaptec1	0.0216	0.0228	0.0236	0.0290	0.0281	0.0281
adaptec2	0.0218	0.0221	0.0215	0.0258	0.0264	0.0267
adaptec3	0.0182	0.0184	0.0184	0.0173	0.0179	0.0179
adaptec4	0.0201	0.0201	0.0202	0.0218	0.0219	0.0219
adaptec5	0.0239	0.0241	0.0234	0.0251	0.0251	0.0243
newblue1	0.0176	0.0183	0.0185	0.0199	0.0205	0.0205
newblue2	0.0147	0.0147	0.0147	0.0217	0.0223	0.0221
Summary	1.000	1.018	1.019	1.000	1.013	1.007

Next we experimentally support the claim in Section 1 that IPD gradient is not a good metric for CMP variations. As suggested in [2], IPD gradient of a tile t_{ij} is defined as $d_{ij} - \bar{d_{ij}}$, where $\bar{d_{ij}}$ is the average IPD of tiles adjacent to t_{ij} (including t_{ij}). Then we compute the quadratic mean of IPD gradients of all tiles as the IPD gradient of a layout:

$$\mathcal{G} = \sqrt{\frac{1}{|Q|} \sum_{t_{ij} \in Q} (d_{ij} - \bar{d_{ij}})^2}, \tag{23}$$

where Q is the set of all the tiles in the layout. Table 5 shows the comparison of IPD gradient, $\mathcal{G}$, among NoCMP, YaCMP and MaxEPD. From the table, we can see that for most circuits, the layouts with the smallest IPD gradients do not have the smallest $\mathrm{ran}(z)$ and D. On average, the layouts obtained by NoCMP have the smallest IPD gradients but the largest $\mathrm{ran}(z)$ and D. In sum, it

is clear that the IPD gradient is not a good indicator for $\mathrm{ran}(z)$ and D.

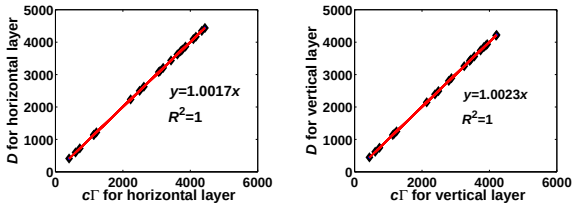


Figure 4: Linear fitting results for $c\Gamma$ vs. D.

Finally, we experimentally support the claim from Section 5.1 that (11) holds under the *dummy fill assumptions*. For all the dummy filling experiments we perform, we find that μ and ν are negligible compared to Γ. For example, for the dummy filling solution for the horizontal layer for circuit newblue2 with MaxEPD algorithm, $(\mu+\nu) : \Gamma = 1 : 483$. Fig. 4 shows the fitting results for dataset $(c\Gamma, D)$ obtained in all the dummy filling experiments, using a linear regression model, "$y = ax$", showing that $D \approx c\Gamma$ with $R^2 = 1$.

7. REFERENCES

[1] Y.-J. Chang et al. NTHU-Route 2.0: a fast and stable global router. In *Proc. ICCAD*, pages 338–343, 2008.

[2] H.-Y. Chen et al. A novel wire-density-driven full-chip routing system for CMP variation control. *IEEE TCAD*, 28(2):193–206, 2009.

[3] M. Cho et al. Wire density driven global routing for CMP variation and timing. In *Proc. ICCAD*, pages 487–492, 2006.

[4] C. Chu and Y. C. Wong. Fast and accurate rectilinear steiner minimal tree algorithm for VLSI design. In *Proc. ISPD*, pages 28–35, 2005.

[5] ILOG CPLEX 11.210. `http://www.ilog.com/products/cplex/`.

[6] ISPD 2007 Global Routing Contest. `http://www.sigda.org/ispd2007/rcontest/`.

[7] ISPD 2008 Global Routing Contest. `http://www.ispd.cc/slides/ispd2008-files/S7-3.pdf`.

[8] Y. Jia et al. Full-chip routing system for reducing Cu CMP & ECP variation. In *Proc. SBCCI*, pages 10–15, 2008.

[9] A. B. Kahng et al. Filling and slotting: analysis and algorithms. In *Proc. ISPD*, pages 95–102, 1998.

[10] A. B. Kahng and K. Samadi. CMP fill synthesis: A survey of recent studies. *IEEE TCAD*, 27(1):3–19, 2008.

[11] T. H. Lee and T. C. Wang. Congestion-constrained layer assignment for via minimization in global routing. *IEEE TCAD*, 27(9):1643–1656, 2008.

[12] K. S. Li et al. Multilevel full-chip routing with testability and yield enhancement. In *Proc. SLIP*, pages 29–36, 2005.

[13] G. Nam et al. The ISPD global routing benchmark suite. In *Proc. ISPD*, pages 156–159, 2008.

[14] D. O. Ouma et al. Characterization and modeling of oxide chemical-mechanical polishing using planarization length and pattern density concepts. *IEEE TSM*, 15(2):232–244, 2002.

[15] M. Pan and C. Chu. FastRoute: a step to integrate global routing into placement. In *Proc. ICCAD*, pages 464–471, 2006.

[16] R. Tian et al. Model-based dummy feature placement for oxide chemical-mechanical polishing manufacturability. *IEEE TCAD*, 20(7):902–910, 2001.

[17] H. Yao et al. CMP-aware maze routing algorithm for yield enhancement. In *Proc. ISVLSI*, pages 239–244, 2007.

APPENDIX

PROOF OF LEMMA 1: From (3), we have

$$\sum_{\text{all } t_{ij}} \rho_{ij} = \sum_{\text{all } t_{ij}} \left(\sum_{n_1=-l}^{l} \sum_{n_2=-l}^{l} \left(d_{i+n_1,j+n_2} f(n_1,n_2) \right) \right)$$
$$= f(-l,-l) \sum_{\text{all } t_{ij}} d_{i-l,j-l} + \cdots + f(l,l) \sum_{\text{all } t_{ij}} d_{i+l,j+l}.$$

For circular convolution, $\sum_{\text{all } t_{ij}} d_{i+p,j+q} = \sum_{\text{all } t_{ij}} d_{ij}, \forall p,q \in \mathbb{Z}$. Therefore, $\sum_{\text{all } t_{ij}} \rho_{ij} = f(-l,-l) \sum_{\text{all } t_{ij}} d_{ij} + f(-l,-l+1) \sum_{\text{all } t_{ij}} d_{ij} + \cdots + f(l,l) \sum_{\text{all } t_{ij}} d_{ij} = b \sum_{\text{all } t_{ij}} d_{ij}$. □

PROOF OF THEOREM 1: By definition, $D = \sum_{t_{ij} \in Q} (d'_{ij} - d_{ij})$, where $Q = Q_0 \cup Q_1$. By Lemma 1 and considering $Q_0 \cap Q_1 = \phi$,

$$\begin{aligned} D &= \frac{1}{b} \sum_{t_{ij} \in Q} (\rho'_{ij} - \rho_{ij}) \\ &= \frac{1}{b} \left(\sum_{t_{ij} \in Q_1} (\rho'_{ij} - \rho_{ij}) + \sum_{t_{ij} \in Q_0} (\rho'_{ij} - \rho_{ij}) \right) \\ &= \frac{1}{b} \left(\sum_{t_{ij} \in Q_1} (\rho_U - \rho_{ij}) + \sum_{t_{ij} \in Q_1} (\rho'_{ij} - \rho_U) + \mu \right) \\ &= c\,(\Gamma + \mu + \nu)\,. \end{aligned}$$ □

PROOF OF THEOREM 2: We first prove part (a). Suppose that the changes from $\mathcal{S}$ to $\bar{\mathcal{S}}$ due to routing through e are: ρ_{ij} changes to $\bar{\rho}_{ij}$, ρ_H changes to $\bar{\rho}_H$, ρ_U changes to $\bar{\rho}_U$, Q_1 changes to $\bar{Q}_1$, Q_0 changes to $\bar{Q}_0$, Γ changes to $\bar{\Gamma}$. Then $\Delta\rho_H = \bar{\rho}_H - \rho_H$. Assume ϵ for the two routing solutions is the same. Then $\bar{\rho}_U = \rho_U + \Delta\rho_H$, $\bar{\Gamma} = \sum_{t_{ij} \in \bar{Q}_1} (\bar{\rho}_U - \bar{\rho}_{ij})$. Then $\Delta\Gamma = \bar{\Gamma} - \Gamma$.

Let $Q_1^+ = \{t_{ij} | t_{ij} \notin Q_1 \text{ and } t_{ij} \in \bar{Q}_1\}$, $Q_1^- = \{t_{ij} | t_{ij} \in Q_1 \text{ and } t_{ij} \notin \bar{Q}_1\}$. Then $\bar{Q}_1 = Q_1 + Q_1^+ - Q_1^-$. Then

$$\begin{aligned} \bar{\Gamma} &= \sum_{t_{ij} \in \bar{Q}_1} (\bar{\rho}_U - \bar{\rho}_{ij}) \\ &= \sum_{t_{ij} \in Q_1} (\bar{\rho}_U - \bar{\rho}_{ij}) + \sum_{t_{ij} \in Q_1^+} (\bar{\rho}_U - \bar{\rho}_{ij}) - \sum_{t_{ij} \in Q_1^-} (\bar{\rho}_U - \bar{\rho}_{ij}). \end{aligned}$$

By the definition of Q_1^-, we can know that $\bar{\rho}_{ij} \geq \bar{\rho}_U, \forall\ t_{ij} \in Q_1^-$, and then $\sum_{t_{ij} \in Q_1^-} (\bar{\rho}_U - \bar{\rho}_{ij}) \leq 0$. Similarly, by the definition of Q_1^+, $\sum_{t_{ij} \in Q_1^+} (\bar{\rho}_U - \bar{\rho}_{ij}) \geq 0$. Note that Q_1^+ can be empty if ρ_{ij} of tile t_{ij}, $\forall\ t_{ij} \in \bar{Q}_0$, is no less than $\bar{\rho}_U$. Then

$$\begin{aligned} \bar{\Gamma} &\geq \sum_{t_{ij} \in Q_1} (\bar{\rho}_U - \bar{\rho}_{ij}), \\ \Delta\Gamma = \bar{\Gamma} - \Gamma &\geq \sum_{t_{ij} \in Q_1} (\bar{\rho}_U - \bar{\rho}_{ij}) - \sum_{t_{ij} \in Q_1} (\rho_U - \rho_{ij}) \\ &= \sum_{t_{ij} \in Q_1} \Delta\rho_H - \sum_{t_{ij} \in Q_1} (\bar{\rho}_{ij} - \rho_{ij}). \end{aligned}$$

Let t_e be the tile associated with edge e and W_e be the weighting window of t_e. Note that only ρ_{ij} of tile $t_{ij} \in W_e$ can increase after routing through edge e. Let $W_1 = Q_1 \cap W_e$. Since $W_1 \subseteq W_e$ and $\bar{\rho}_{ij} \geq \rho_{ij}$, we have

$$\begin{aligned} \Delta\Gamma &\geq \sum_{t_{ij} \in Q_1} \Delta\rho_H - \sum_{t_{ij} \in W_1} (\bar{\rho}_{ij} - \rho_{ij}) \\ &\geq \sum_{t_{ij} \in Q_1} \Delta\rho_H - \sum_{t_{ij} \in W_e} (\bar{\rho}_{ij} - \rho_{ij}). \end{aligned} \tag{24}$$

Since the increase of ρ_{ij} of $t_{ij} \in W_e$ is due to the increase in the IPD of t_e by Δ_d^M, we have

$$\sum_{t_{ij} \in W_e} (\bar{\rho}_{ij} - \rho_{ij}) = \Delta_d^M \sum_{i=-l}^{l} \sum_{j=-l}^{l} f(i,j) = b\Delta_d^M. \tag{25}$$

By (24) and (25), we have $\Delta\Gamma \geq |Q_1|\Delta\rho_H - b\Delta_d^M$.

Note that the conclusion holds when $\Delta\rho_H \geq 0$, which is easy to see from the proof itself.

For part (b), we use the same symbols defined above. Using the conditions $\Delta\rho_H = 0$, $\bar{Q}_1 = Q_1$ and $W_e \subseteq Q_1$, similar to the proof above, we have:

$$\Delta\Gamma = -\sum_{t_{ij} \in Q_1} (\bar{\rho}_{ij} - \rho_{ij}) = -\sum_{t_{ij} \in W_e} (\bar{\rho}_{ij} - \rho_{ij}) = -b\Delta_d^M.$$

□