Performance-Impact Limited Area Fill Synthesis *

Yu Chen, Puneet Gupta†, and Andrew B. Kahng‡

Computer Science Dept., UCLA, Los Angeles, CA 90095-1596
†Electrical and Computer Engineering Dept., UCSD, La Jolla, CA 92093-0114
‡CSE and ECE Departments, UCSD, La Jolla, CA 92093-0114

ABSTRACT

Chemical-mechanical planarization (CMP) and other manufacturing steps in very deep-submicron VLSI have varying effects on device and interconnect features, depending on the local layout density. To improve manufacturability and performance predictability, area fill features are inserted into the layout to improve uniformity with respect to density criteria. However, the performance impact of area fill insertion is not considered by any fill method in the literature. In this paper, we first review and develop estimates for capacitance and timing overhead of area fill insertion. We then give the first formulation of the Performance Impact Limited Fill (PIL-Fill) problem, and describe three practical solution approaches based on Integer Linear Programming (ILP-I and ILP-II) and the Greedy method. We test our methods on two layout testcases obtained from industry. Compared with the normal fill method,[3] our ILP-II method achieves between 25% and 90% reduction in terms of total *weighted edge delay* (roughly, a measure of sum of node slacks) impact, while maintaining identical quality of the layout density control.

Keywords: Area Fill, Performance Impact, Coupling Capacitance, Signal Delay

1. INTRODUCTION

Chemical-mechanical planarization (CMP) and other manufacturing steps in nanometer-scale VLSI processes have varying effects on device and interconnect features, depending on local attributes of the layout. To improve manufacturability and performance predictability, foundry rules require that a layout be made uniform with respect to prescribed density criteria, through insertion of *area fill* ("dummy") geometries.

All existing methods for synthesis of area fill are based on discretization[34]: the layout is partitioned into *tiles*, and filling constraints or objectives (e.g., minimizing the maximum variation in feature area content) are enforced for square *windows* that each consists of $r \times r$ tiles. In practice, then, layout density control is achieved by enforcing density bounds in a finite set of windows. Invoking terminology from previous literature, we say that the foundry rules and EDA tools (physical verification and layout) attempt to enforce density bounds within r^2 overlapping *fixed dissections*, where r determines the "phase shift" w/r by which the dissections are offset from each other. The resulting *fixed r-dissection* (see Figure 1) partitions the $n \times n$ layout into tiles T_{ij}, then covers the layout by $w \times w$-windows W_{ij}, $i, j = 1, \ldots, \frac{nr}{w} - 1$, such that each window W_{ij} consists of r^2 tiles T_{kl}, $k = i, \ldots, i + r - 1$, $l = j, \ldots, j + r - 1$.

While area fill feature insertion can significantly reduce layout density variation, it can also change interconnect signal delay and crosstalk by changing coupling capacitance. These changes can be harmful to timing closure flows, especially since fill is typically added as a physical verification or even post-GDSII (at the foundry) step. Therefore, in addition to satisfying density requirements, dummy fill insertion should also minimize *performance impact* associated with metal fill. However, the issues associated with capacitance and area fill are complex. Currently, metrological methodologies are used to find out the "best" choice of buffer

* This research was supported by a grant from Cadence Design Systems, Inc., and by the MARCO/DARPA Gigascale Silicon Research Center.

Corresponding author: Yu Chen. Emails: Yu Chen (yuchen@cs.ucla.edu), Puneet Gupta (puneet@ucsd.edu), Andrew Kahng (abk@ucsd.edu).

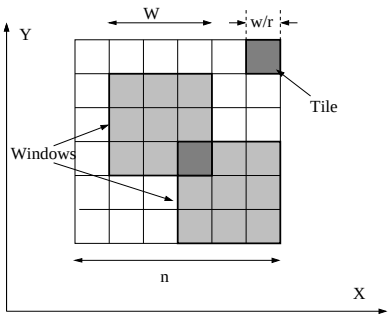

Figure 1. In the fixed r-dissection framework, the n-by-n layout is partitioned by r^2 (here, $r = 3$) distinct overlapping dissections with window size $w \times w$. This induces $\frac{nr}{w} \times \frac{nr}{w}$ *tiles*. Each dark-bordered $w \times w$ window consists of r^2 tiles.

distance, dummy fill types (grounded versus floating), and dummy fill patterns. There is no existing published work on performance-driven area fill synthesis.[†] In this paper, our contributions include:

- a new *PIL-Fill* problem formulation for performance-impact limited area fill synthesis;
- practical integer linear programming formulations, along with a greedy method, for the Minimum Delay, Fill-Constrained variant of the PIL-Fill problem; and
- experimental comparisons of our three approaches, confirming the advantages of our work over previous methods (up to 90% reductions in delay impact compared with the normal fill methods[3]), and identifying at least one practical method for deployment.

In this work, we assume that area fill consists of squares of floating fill; we seek a fill placement with minimum delay impact of fill insertion. In the next section, we review related works in the PIL-Fill domain. In Section 3, we briefly review interconnect capacitance estimation models, and describe our simplified capacitance impact and delay impact model for floating fill. Section 4 formulates the PIL-Fill problem, and solution approaches are given in Section 5. Section 6 gives experimental results and we conclude in Section 7.

2. RELATED WORK

According to Stine et al.'s paper,[11] to minimize the increase in interconnect capacitance that results from area fill, (i) the total amount of added fill should be minimized, (ii) the linewidth of the fill pattern should be minimized, (iii) the spacing between fill lines should be maximized, and (iv) the buffer distance should be maximized. Unfortunately, these guidelines are rather generic. We observe that restricting the amount of dummy fill and increasing the buffer distance has the unwanted effect of limiting the possible improvements in uniformity achieved by fill insertion. Furthermore, such guidelines are not precisely matched to the relevant underlying criteria: e.g., the capacitance minimization objective is oblivious to the delay and timing slack impact of the added capacitance. While no work has (in our opinion) yet addressed the PIL-Fill problem, in the remainder of this section we review two related works.

Work at Motorola by Grobman et al.[8] points out that the main parameters to influence the change in interconnect capacitance due to fill insertion are feature ("block") sizes and proximity to interconnect lines. The larger the size of the block, the larger the consequent interaction between interconnect lines. Similarly, the closer blocks are to interconnect lines, the stronger their interaction will be.

Grobman et al.[8] consider several structures that are expected to represent the most profound effects. In one limiting case, dense lines effectively do not suffer much from floating fill placement above and below, since their capacitance is dominated by coupling to neighbors. On the other hand, when interconnect lines are

[†]Although this concept has been recently mentioned in some startup web sites,[10, 13, 14] no details of functionality are given.

more sparsely situated, floating fill has greater performance impact. The importance of dummy fill size is also examined: large block shapes more effectively transmit local effects to their extent, and hence if filling is to be performed over critical paths, use of smaller fill blocks with the same filling density helps limit the increase of interconnect capacitance.

Work at MIT Microsystems Technology Laboratories[11] proposes a rule-based area fill methodology. To minimize the added interconnect capacitance resulting from fill, a dummy fill design rule is designed by modeling the effects on interconnect capacitance of different design rules (while satisfying the density requirement). Three canonical parameters are considered in design rules: buffer distance (buf), block width (w), the block space (s). Effects of fill on interconnect capacitance are calculated from the canonical parameters as well as line width and spacing. The calculation result is coupled with the minimum pattern density goals to obtain an optimized dummy fill design rule. It is important to note that the MIT methodology yields only a *rule*: the fill insertion is not driven by any context (e.g., per-net or per-wire segment delay or slack considerations).

3. CAPACITANCE AND DELAY MODELS

We now review basic interconnect capacitance models and provide simplified expressions to model capacitance and delay impact of fill insertion. Works on multilayer interconnect capacitance extraction include 1-D, 2-D, 2.5-D and 3-D analytic models.[1, 2, 5, 6, 12] For example, the interconnect capacitance[1] at each circuit node is calculated via a model consisting of three conducting layers over the substrate treated as a (ground potential) reference plane. In general, the capacitance of interest at any node consists of three components, (i) *overlap (area) capacitance*, C_a, formed by the surface overlap (in two dimensions) of two conductors; (ii) *lateral coupling capacitance*, C_{lt}, between two parallel conductors on the same plane; and (iii) *fringe capacitance*, C_{fr}, that represents coupling between two conductors on different planes. In other words, the interconnect capacitance at any node is given by

$$C_t = C_a + C_{lt} + C_{fr} \tag{1}$$

Overlap and fringing capacitance of active (switching) lines are not significantly affected by the insertion of small floating dummy features[1]; we hence mainly consider the impact of area fill on the lateral coupling capacitance between active lines.

A typical fill insertion approach is to grid the layout into sites according to the fill feature size and design rules, then insert the fill features into the slack sites to satisfy the density requirements. To make the following discussion clear, we group the fill features between two parallel active lines as:

- *a row of dummy features* is a line of dummy features which is parallel to the active lines; and
- *a column of dummy features* is a line of dummy features which is perpendicular to the active lines.

To estimate area fill impact on active line delay, we focus on the capacitance increment in the active line due to the fill. In Figure 2(A), the total capacitance of an active line before area fill is inserted can be written as

$$C_{orig} = C_B \cdot l \tag{2}$$

$$C_B = \frac{\epsilon_0 \epsilon_r a}{d} \tag{3}$$

where C_B is the per-unit length capacitance between the active line and its neighboring active line, l is the overlap length of the two active lines, ϵ_0 is permittivity of free space, ϵ_r is the relative permittivity of the material between the two conductors, and a is the overlapping area between them.

For the general case (with two rows of dummy fills) in Figure 2(A), the total capacitance between two active lines is

$$C_{fill} = (\frac{1}{1/C_A + 1/C_C + 1/C_A}) \cdot w \cdot k + C_B \cdot (l - w \cdot k) \tag{4}$$

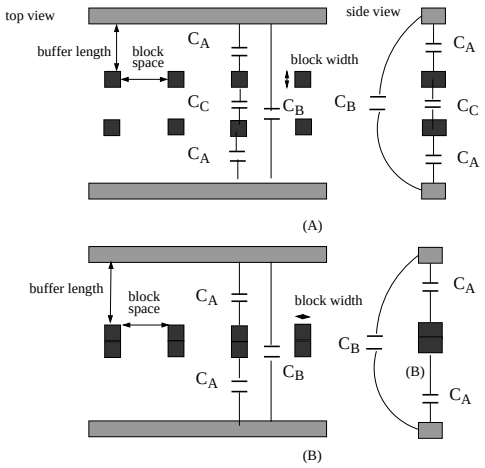

Figure 2: Example configurations of floating dummy fill.

where C_A is the capacitance between the dummy feature and the active line, and C_C is the capacitance between the dummy features. In this equation, w is the dummy feature width, s is the space between dummy features, and k is the number of dummy features between the two active lines. Note that we assume that the floating dummy features have no effect on C_B due to their small size.

To simplify the estimation, we use a simple parallel plate capacitance model. We can then approximate the impact of two rows of dummy features by making one combined row of dummy features, as shown in Figure 2(B). Generalizing to m rows of dummy features, we obtain the following estimate of per-unit coupling capacitance between two active lines where there are m dummy features in a column between them as:

$$C_{A'} = f(m) = \frac{\epsilon_0 \cdot \epsilon_r \cdot a}{d - m \cdot w} \tag{5}$$

$$\approx \epsilon_0 \cdot \epsilon_r \cdot a \cdot (1 + \frac{m \cdot w}{d})/d$$

$$= C_B + \frac{\epsilon_0 \cdot \epsilon_r \cdot a \cdot m \cdot w}{d^2} \tag{6}$$

Here, we model m dummy features in one column between two active lines as a single metal block with length $m \cdot w$ between the lines. When $w \ll d$, we can further simplify the calculation as a linear one (see Equation (6)), where $\frac{\epsilon_0 \cdot \epsilon_r \cdot a \cdot m \cdot w}{d^2}$ is the incremental capacitance due to dummy feature insertion.

So, the total capacitance between two active lines can be estimated as:

$$C_{fill} = C_{A'} \cdot w \cdot k + C_B \cdot (l - w \cdot k) \tag{7}$$

With respect to interconnect delay, our discussion below will use the Elmore delay model to estimate total delay increase due to area fill. Elmore delay[7] of a cascaded N-stage RC chain (Figure 3) is given by

$$\tau_N = \sum_{i=1}^{N} R_i \sum_{j=i}^{N} C_j = \sum_{i=1}^{N} C_i \sum_{j=1}^{i} R_j \tag{8}$$

So, each node j on the chain contributes to τ the product of the capacitance at node j and the total resistance between j and the source node. If the capacitance at node i increases by ΔC_i, the increment of Elmore delay at any node k below node i is

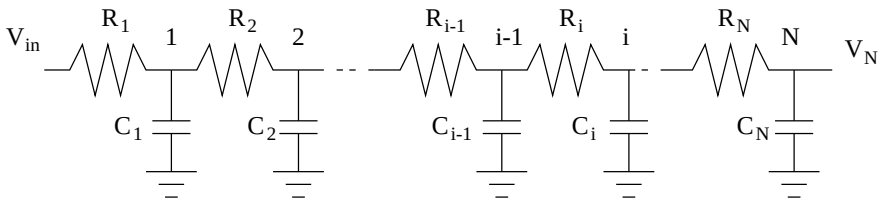

Figure 3: Segmented RC line model.

$$\Delta\tau_k = \Delta C_i \sum_{j=1}^{i} R_j \tag{9}$$

From Equation (8), we know that Elmore delay enjoys an additivity property with respect to capacitance along any source-sink path. That is, if we add the coupling capacitance C_x at position x, the delay of the nodes after the position x will increase by $C_x \cdot R_x$. Here, R_x is constant, and equal to the total resistance between the source and the position x (below, we call this an *entry resistance*, i.e., an "upstream" resistance).

4. PROBLEM FORMULATIONS

Performance-impact limited area fill synthesis has two objectives:

- minimizing the layout density variation due to CMP planarization; and
- minimizing the dummy features' impact on circuit performance (e.g., signal delay and timing slack).

It is difficult to satisfy the two objectives simultaneously. Practical approaches will tend to optimize one objective while transforming the other into constraints. In this section, we propose a performance-impact limited area fill problem formulation (PIL-Fill) in which the objective is to minimize delay impact, subject to a constraint of prescribed amounts of fill in every tile region of the layout. We call this a *minimum delay with fill constraint*, or MDFC, formulation.[‡]

The MDFC PIL-Fill problem can be stated as follows. *Given a fixed-dissection routed layout and the design rule for floating square fill features, insert a* prescribed *amount of fill in each tile such that the performance impact (i.e., the total increase in wire segment delay) is minimized.*

Since each tile can be considered independently in the fixed-dissection layout, we may reformulate the MDFC PIL-Fill problem on a per-tile basis. In other words, for *each* tile the following optimization is separately performed.

Given tile T, a prescribed total area of fill features to be added into T, a size for each fill feature, a set of slack sites *(i.e., sites available for fill insertion) in T per the design rules for floating square fill, and the direction of current flow and the per-unit length resistance for each interconnect segment in T, insert fill features into in T such that total impact on delay is minimized.*

A weakness of this formulation is that we minimize the total delay impact *independently* in each tile. We are *not* able at this point to ensure, e.g., that total impact on every timing path is less than available positive timing slack. (Possible methodologies, including the use of budgeted capacitances that are typically available during timing-driven synthesis, place and route, are noted below.) For now, we will focus on the MDFC PIL-Fill formulation per tile, using the capacitance approximations given above[§] and the Elmore delay model. Under

[‡]We have also studied a *minimum variation with delay constraint* formulation, but it is less tractable to optimization heuristics and we do not discuss it here.

[§]These are essentially the same as those used in the MIT approach.[11]

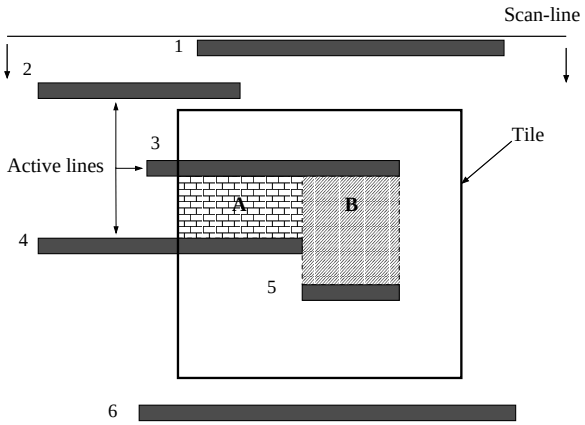

Figure 4: SlackColumn-I definition: Scan-line and slack blocks between pairs of active lines within tile.

the Elmore delay model, the impact of each wire segment delay on the total sink delay of the routing net is found by multiplying by the number of downstream sinks. Thus, we define the *weight* of an active line l as

$$W_l \equiv \text{number of downstream sinks}$$

which allows us to directly minimize total sink delay impact over all nets in a given tile.[¶]

5. APPROACHES FOR MDFC PIL-FILL

We now present three methods for the MDFC PIL-Fill problem, after developing necessary preliminary material on *slack site columns* and the scan-line approach.

5.1. Slack Site Columns and Scan-Line Algorithm

The key computational geometry task in solving MDFC PIL-Fill is to find all pairs of parallel active line segments, and the *slack site columns* (i.e., columns of available sites for fill features) between each pair of active lines. We define the *size of a slack site column* as the number of sites in the column. Without loss of generality, we assume that the routing direction is horizontal on the selected layer. The column index k of the sites in the slack site column can used as the index of the column between two active lines. We discuss three distinct definitions of *slack site columns* within a tile; these definitions are ordered by increasing accuracy in how they capture the capacitance impact of fill on active lines.

- **SlackColumn-I:** The simplest definition captures only slack site columns between active lines within the tile. Figure 4 shows two slack blocks A and B between the active lines in the given tile. To find such slack columns, only the active lines intersecting with the tile need to be scanned for each tile. However, with this definition, the remaining slack space in the tile cannot be used during fill insertion, which causes problems when the total size of slack columns is less than the required number of fill features.
- **SlackColumn-II:** A more accurate definition captures slack site columns between the active lines, between the active line and tile boundary, and between tile boundaries. Figure 5 shows six such slack blocks in the tile. Among them, slack columns in block B have no associated active lines in the tile. Obviously, the drawback of this slack site column definition is that fill features will be inserted into the slack columns without consideration of associated active lines (e.g., outside the tile, with respect to block B), and this causes inaccuracy with respect to the minimum delay impact objective.

[¶]This objective, which is correlated with total impact on sink actual arrival times, moves us closer to the ideal of being timing-slack driven.

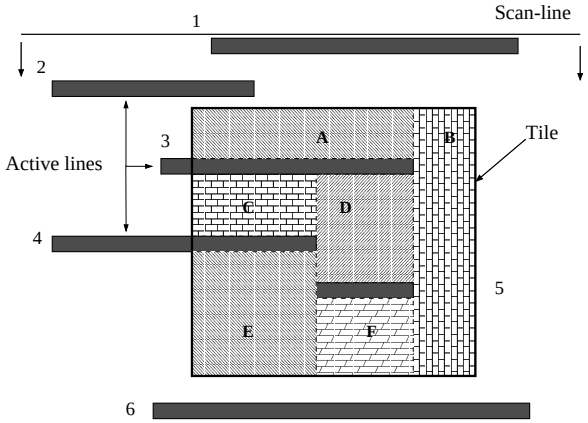

Figure 5. SlackColumn-II definition: Illustration of scan-line and slack blocks between pairs of active lines, active line and tile boundary, and tile boundaries within tile.

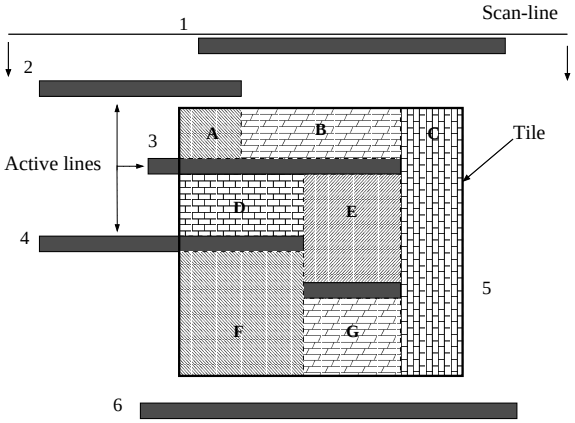

Figure 6. SlackColumn-III definition: Illustration of scan-line and slack blocks within tile between pairs of active lines in adjacent tiles.

- **SlackColumn-III:** Finally, we can define slack site columns to capture the cases of slack sites between the active lines in adjacent tiles as well as between the active lines and layout boundary. Figure 6 shows seven such slack blocks in the tile; the columns in each block will be accounted for in estimating capacitance increase in the tile. This is the most accurate definition of slack slack column since the possible impact of fill features at any position on the layout will be considered. For example, the fill features located in the slack block C in Figure 6 will affect the coupling capacitance on active lines 1 and 6, while those in the slack block B in Figure 5 do not have their impacts on any active lines captured.

To find the SlackColumn-III columns in the layout, we first obtain the position of each active line. After sorting according to y-coordinates (for horizontal routing direction) or x-coordinates (for vertical routing direction), we scan the whole layout from the bottom boundary (for horizontal routing direction) or from the left boundary (for vertical routing direction) to find the slack columns between active lines or between boundary and active line. Finally, we calculate the overlapping area of each slack column in each tile intersecting with it, as well as the list of active lines intersecting with each tile. The algorithm is described in detail in Figure 7.

Scan-Line Algorithm to Compute Fill Slack Column
Input: a design-rule correct layout; tile size t; site size s; list of active lines in layout AL
Output: a list of slack columns $Cols$ on the layout; lists of slack columns $Cols_{ij}$ intersecting with each tile T_{ij}; lists of active lines AL_{ij} intersecting with each tile
1. Partition the layout into $m \times n$ tiles and $k \times l$ sites
2. Sort the list of active lines according to its Y coordinates
3. Initialize the list of slack columns $Cols$
4. Create k empty slack columns starting from the bottom boundary
5. **While** there is active line left in AL **Do**
6. Get the active line with smallest Y coordinate in list
7. **For** each slack column intersecting with the active line **Do**
8. End the slack column at the active line
9. **If** the size of slack column is larger than 0 **Then**
10. Add the slack column into $Cols$
11. **Else** Ignore the slack column
12. Create a new one starting from the active line
13. Delete active line from the list
14.**For** all slack columns ended at the top boundary **Do**
15. **If** the size of slack column is larger than 0 **Then**
16. Add the slack column into $Cols$
17.Initialize lists of slack columns $Cols_{ij}$ and lists of active lines AL_{ij}
18.**For** each slack column **Do**
19. **For** each tile T_{ij} overlapping with the slack column **Do**
20. Calculate the number of sites in slack column within the tile T_{ij}
21. Add the slack column into $Cols_{ij}$
22. Add the correlated active line(s) into AL_{ij}

Figure 7: Scan-line algorithm to find fill slack columns on given layer (assuming horizontal routing direction).

5.2. Integer Linear Programming Approach I

In our flow, we calculate post-routing interconnect delay after obtaining routing information from a DEF file. From the analysis in Section 3, we know that the columns of dummy features have the additivity property with respect to coupling capacitance, and we can approximate the coupling capacitance of m dummy features in one column by a linear function (6). Without loss of generality, we assume the routing direction on the layer is horizontal, and we also ignore any wrong-direction routing. The PIL-Fill problem is then captured by an Integer Linear Programming formulation. We first make the following definitions.

- $W_l \equiv$ weight of active line l;
- $C_k \equiv$ size (capacity) of feasible slack site column k for dummy features within the tile;
- $m_k \equiv$ number of dummy features inserted in column C_k;
- $Cap_k \equiv$ incremental capacitance caused by the m_k dummy features in column C_k, calculated according to Equation 6;
- $\tau_l \equiv$ total delay increment on active line l due to the insertion of dummy features along it in the tile;
- $R_l \equiv$ total (upstream) resistance of path from the source node to the entry point of active line l into the tile; and
- $r_l \equiv$ per-unit resistance of active line l.

Minimize:

$$\sum_{l=1}^{L} W_l \cdot \Delta\tau_l \qquad \text{over all active lines} \tag{10}$$

Subject to:

$$F = \sum m_k \quad \text{over all slack columns in tile} \tag{11}$$

$$Cap_k = \frac{\epsilon_0 \cdot \epsilon_r \cdot a \cdot w}{{d_k}^2} \cdot m_k \quad \text{for each slack column} \tag{12}$$

$$\Delta\tau_l = \sum_k Cap_k \cdot (R_l + \sum_{s=p}^{k} r_l) \quad \text{over all slack columns along each active line} \tag{13}$$

$$\text{Integer:} \quad 0 \leq m_k \leq C_k \tag{14}$$

- The objective function (10) implies that we minimize the weighted incremental Elmore delay caused by dummy feature insertions. L is the total number of active lines in the tile.
- Constraints (11) imply that the total number of covered slack sites is equal to the number of dummy features.
- Constraints (12) are used to capture the incremental capacitance caused by m_k dummy features in column k between each pair of active lines. Here, a is the overlapping area between two active lines, d_k is the distance between them, and w is the dummy feature width.
- Constraints (13) are used to capture the total Elmore delay increment due to dummy feature insertions in all slack columns along the active line l in the tile. $(R_l + \sum_{s=p}^{k} r_l)$ is the total resistance between the source and the position k on the active line l in the tile. p is the x-coordinate of the left-most point of the active line in the tile, k is the x-coordinate of slack column k.
- Constraints (14) imply that the number of covered (i.e., used) slack sites in any column should be less than the column size (capacity).

5.3. Integer Linear Programming Approach II

In the previous subsection, we used the linear approximation for coupling capacitance between two active lines after dummy fill insertion. This is not accurate when the dummy feature width is not substantially less than the distance between the two active lines. Since (i) all dummy features have the same shape, (ii) the potential number of dummy fill features (and their positions, given the fixed-dissection layout) in each slack column is limited, (iii) the size of any slack column is also limited, and (iv) the other parameters (ϵ_o, ϵ_r, and d) in Equation (5) are constant for each pair of active lines, we can pre-build a lookup table for $f(n, d)$ for each pair of active lines separated by distance d. Based on the lookup table, a more accurate ILP formulation can be given. We add the following definition to our terminology.

- $m_{k_n} \equiv$ auxiliary boolean variable:

$$m_{k_n} = \begin{cases} 1 & \textit{if } m_k = n \\ 0 & \textit{otherwise} \end{cases} \tag{15}$$

Minimize:

$$\sum_{l=1}^{L} W_l \cdot \Delta\tau_l \quad \text{over all active lines} \tag{16}$$

Subject to:

$$F = \sum m_k \quad \text{over all slack columns} \tag{17}$$

$$m_k = \sum_{n=1}^{C_k} n \cdot m_{k_n} \quad \text{for each slack column} \tag{18}$$

$$\sum_{n=1}^{C_k} m_{k_n} = 1 \quad \text{for each slack column} \tag{19}$$

$$Cap_k = \sum_{n=1}^{C_k} f(n, d_k) \cdot m_{k_n} \quad \text{for each slack column} \tag{20}$$

$$\Delta\tau_l = \sum_{k} Cap_k \cdot (R_l + \sum_{s=p}^{k} r_l) \quad \text{over all slack columns along each active line} \tag{21}$$

$$\text{Integer:} \quad 0 \le m_k \le C_k \tag{22}$$

$$\text{Binary:} \quad m_{k_n} \tag{23}$$

- Constraints (18), (19), (20), and (23) replace constraints (12) in the ILP-I formulation.
- Constraints (18 and 19) imply that m_k can only be assigned one value from 1 to C_k.
- Constraints (20) is the equation for coupling capacitance based on the lookup table. Here, $f(n \cdot m_{k_n})$ is the constant value from the pre-built lookup table.
- Constraints (21) are used to capture the total Elmore delay increment due to dummy feature insertions in all slack columns along the active line l in the tile. $(R_l + \sum_{s=p}^{k} r_l)$ is the total resistance between the source and the position k on the active line l in the tile. p is the x-coordinate of the left-most point of the active line in the tile, k is the x-coordinate of slack column k.

5.4. Greedy Method

From Equation (13), the impact on delay due to the dummy features is dependent on the total resistance between the source and the current node. Our final algorithmic approach is to greedily insert dummy features along active line segments where the incremental delay is minimum. This greedy approach is described in Figure 8.[‖]

6. COMPUTATIONAL EXPERIENCE

We have tested our proposed algorithms using two layout test cases, denoted T1 and T2, obtained from industry sources. Each of the test cases was obtained in LEF/DEF format, and "Normal" fill was synthesized using the normal fill method[3] according to the parameters shown in the leftmost column of Table 1.[**].

In Table 1, we measure the total delay increase on all wire segments due to the "normal" fill method,[3] and due to our three performance-impact limited fill methods. As shown in the table, all total delay increases from the PIL-Fill methods are better than the total delay increase resulting from the normal fill method.[3] Among the PIL-Fill methods, the ILP-II method has the smallest delay increase (e.g, up to 90% reduction in non-weighted total delay increase for case $T1/32/2$, compared to the normal fill result) and its run time is reasonable. The Greedy method is better than the ILP-I method, but not nearly as good as the ILP-II method. The linear approximation used in the ILP-I method is apparently unreasonable, i.e., its loss of accuracy is too great. For example, for cases $T1/32/8$, $T1/20/2$, and $T1/20/4$, the results from the ILP-I method are even worse than the normal fill results. Our experiments also show that the improvement in total delay impact depends on dissection size. As explained above, when the dissection becomes too fine-grain, it becomes harder to consider the total impact of a slack site column since we handle the overlapping tiles separately.

Table 2 shows the results from the weighted performance-impact limited fill methods. Similar to the non-weighted PIL-Fill results, the ILP-II method gives the best solution quality (e.g., up to 93% reduction in weighted total delay increase for case $T1/32/2$, compared to the normal fill method) and retains its practicality.

[‖]As presented, the Greedy algorithm will tend to insert fill close to the active line with minimum resistance. This may lead to worsening of critical path delay and hence cycle time in some pathological cases, compared to random fill insertion. This can be circumvented by placing an upper bound on the added net delay.

[**]Our experimental testbed integrates GDSII Stream and internally-developed geometric processing engines, coded in C++ under Solaris 2.8. We use CPLEX version 7.0 as the integer linear programming solver. All runtimes are CPU seconds on a 300 MHz Sun Ultra-10 with 1 GB of RAM.

Greedy PIL-Fill Algorithm

Input: the design-correct layout; window size w; dissection value r;
the fill pattern (size of fill feature s, gap between fill features g,
and buffer distance from interconnect b)

Output: filled layout minimizing density variation while satisfying
upper bound on coupling capacitance for each net

1. Partition the layout into tiles and sites
2. Run LP/Monte-Carlo[3] to get the number of required fillFeatures ($numRF_{ij}$) for each tile T_{ij}
3. **For** each net N_i in the layout **Do**
4. Find its intersection with each tile T_{ij}
5. Calculate entry resistances $R_l(p,q)$ of N_i in its intersected tiles
6. Find signal directions of N_i in its intersected tiles
7. Run scan-line algorithm (Fig. 7) to get slack site columns in layout
8. **For** each tile T_{ij} **Do**
9. **For** each slack site column k **Do**
10. Find overlapping area of column k in tile T_{ij}
11. Calculate cumulative resistance $\hat{r}_k$ at position k on two neighboring active lines l and l' as: $W_l(R_l(i,j) + \sum_{s=p}^{k} r_l) + W_l'(R_l'(i,j) + \sum_{s=p'}^{k} r_{l'})$
12. Calculate induced coupling capacitances $\hat{Cap}_k$ of column k as in Equation (5) with C_k dummy features
13. Sort all slack columns in the tile according to its corresponding delay as $\hat{r}_k \cdot \hat{Cap}_k$
14. Initialize the number of filled features for tile T_{ij}: $numFF_{ij} = 0$
15. **While** $numFF_{ij} < numRF_{ij}$ **Do**
16. Select slack column C_k with the minimum corresponding delay
17. Insert $\min((numRF_{ij} - numFF_{ij}), C_k)$ dummy features in the slack column
18. Delete the slack column
19. $numFF_{ij} += \min((numRF_{ij} - numFF_{ij}), C_k)$

Figure 8: Greedy PIL-Fill algorithm.

Testcase	**Normal**	**ILP-I**		**ILP-II**		**Greedy**	
T/W/r/	τ	τ	CPU	τ	CPU	τ	CPU
T1/32/2	114.0	88.2	120	12.1	689	91.5	117
T1/32/4	126.8	111.1	136	32.8	525	101.2	125
T1/32/8	95.1	124.7	164	43.0	330	100.2	136
T1/20/2	188.6	233.7	149	46.7	456	186.2	120
T1/20/4	174.4	170.4	169	78.6	480	164.4	142
T1/20/8	124.4	112.4	245	86.6	461	117.5	151
T2/32/2	1070.6	719.8	21	461.5	121	688.9	16
T2/32/4	855.8	520.3	24	212.8	80	517.3	16
T2/32/8	787.5	575.9	35	407.7	77	545.7	18
T2/20/2	1456.5	1120.9	29	542.6	99	1054.1	23
T2/20/4	1265.4	1069.7	35	793.4	82	1062.1	22
T2/20/8	1396.1	1275.4	66	879.2	112	1136.8	27

Table 1. Non-weighted PIL-Fill synthesis. **Notation:** *T/W/r:* testcase / window size / r dissection; *Normal:* normal fill result; *ILP-I:* Integer Linear Programming method I; *ILP-II:* Look-Up Table Integer Linear Programming method; *Greedy:* Greedy method; τ*:* total delay increase (ns); *CPU:* runtime (seconds).

7. CONCLUSIONS AND FUTURE RESEARCH

In this paper, we have developed approximations for capacitance impact of area fill insertion, and given the first formulation for the Performance Impact Limited Fill (PIL-Fill) problem. We have presented two Integer Linear Programming based approaches and a Greedy method; experiments on industry layouts indicate that our PIL-Fill methods can reduce the total delay impact of fill by very significant percentages. The lookup-table based ILP method (ILP-II) has the best performance with respect to the total delay increase - both weighted

Testcase	Normal	ILP-I		ILP-II		Greedy	
T/W/r/	τ	τ	CPU	τ	CPU	τ	CPU
T1/32/2	513.1	264.1	120	34.47	686	230.5	170
T1/32/4	525.1	537.5	136	107.0	539	287.4	148
T1/32/8	1399.4	1378.5	164	278.3	328	316.5	135
T1/20/2	836.5	734.4	148	142.4	439	448.5	177
T1/20/4	781.3	779.4	167	378.6	444	528.4	157
T1/20/8	595.5	577.2	239	391.4	460	519.7	154
T2/32/2	5322.7	2329.8	21	1722.8	122	2217.2	16
T2/32/4	4510.6	1913.6	24	566.9	80	1750.1	16
T2/32/8	4109.4	3225.7	35	1371.7	76	2621.0	17
T2/20/2	6074.9	3457.9	29	1261.2	100	3031.7	23
T2/20/4	6887.4	5569.0	35	2532.3	81	4536.5	22
T2/20/8	8041.9	8841.4	63	6090.4	108	6618.8	26

Table 2: Weighted PIL-Fill synthesis.

and non-weighted - and has practical runtimes.

Our ongoing research is focused on performance-impact limited fill synthesis with given capacitance budgets for each net. This corresponds to the availability of budgeted slacks (translated to budgeted capacitances), which are typically available within synthesis, place and route tools driven by incremental static timing engine. Budgeted slacks (capacitances) eliminate the obstacle of handling full timing paths, and will allow us to evaluate our methods within an integrated layout-manufacturing timing closure flow. Other research addresses alternative PIL-Fill formulations, e.g., wherein an upper bound on timing impact constrains the minimization of layout density variation.

REFERENCES

1. N. D. Arora, K. V. Raol, R. Schumann, and L. M. Richardson, "Modeling and Extraction of Interconnect Capacitances for Multi-layer VLSI Circuits," *IEEE Trans. on Computer-Aided Design* 15(1) (1996), pp. 58-67.
2. E. Barke, "Line-to-Ground Capacitance Calculation for VLSI: A Comparison", *IEEE Trans. on Computer-Aided Design* 7(2) (1988), pp. 295-298.
3. Y. Chen, A. B. Kahng, G. Robins and A. Zelikovsky, "Dummy Fill Synthesis for Uniform Layout Density", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 21(10) (2002), pp. 1132-1147.
4. Y. Chen, A. B. Kahng, G. Robins and A. Zelikovsky, "Smoothness and Uniformity of Filled Layout for VDSM Manufacturability," *Proc. ACM/IEEE International Symposium on Physical Design*, April 2002, pp. 137-142.
5. J. Chern, J. Huang, L. Aldredge, P. Li and P. Yang, "Multilevel Metal Capacitance Models for Interconnect Capacitances", *IEEE Electron Device Lett* EDL-14 (1992), pp. 32-43.
6. J. Cong, L. He, A. B. Kahng, D. Noice, N. Shirali and S. H.-C. Yen, "Analysis and Justification of a Simple, Practical 2 1/2-D Capacitance Extraction Methodology", *Proc. ACM/IEEE Design Automation Conf.*, 1997, pp. 627-632.
7. W. C. Elmore, "The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers", *Journal of Applied Physics* (1948), pp.55-63.
8. W. Grobman, M. Thompson, R. Wang, C. Yuan, R. Tian, and E. Demircan, "Reticle Enhancement Technology: Implications and Challenges for Physical Design ", *Proc. ACM/IEEE Design Automation Conf.*, 2001, pp. 73-78.
9. A. B. Kahng, S. Muddu and E. Sarto, "On Switch Factor Based Analysis of Coupled RC Interconnects", *Proc. ACM/IEEE Design Automation Conf.*, 2000, pp. 79-84.
10. *Praesagus, Inc.*, `http://www.praesagus.com/`
11. B. E. Stine, D. S. Boning et al., "The Physical and Electrical Effects of Metal Fill Patterning Practices for Oxide Chemical Mechanical Polishing Processes", *IEEE Trans. on Electron Devices* 45(3) (1998), pp. 665-679.
12. A. Toulouse, D. Bernard, C. Landrault and P. Nouet "Efficient 3D Modeling for Extraction of Interconnect Capacitances in Deep Submicron Dense Layouts", *Proc. Design Automation and Test Europe (DATE)*, 1999, pp. 576-580.
13. *UbiTech. Inc.*, `http://www.ubitechnology.com/`
14. *XYALIS*, `http://www.xyalis.com/`