3.2

Performance-Impact Limited Area Fill Synthesis *

Yu Chen*, Puneet Gupta†, and Andrew B. Kahng‡
*UCLA Computer Science Dept., 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
yuchen@cs.ucla.edu, puneet@ucsd.edu, abk@ucsd.edu

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 insertions. We then give the first formulations of the Performance Impact Limited Fill (PIL-Fill) problem with the objective of either minimizing total delay impact (MDFC) or maximizing the minimum slack of all nets (MSFC), subject to inserting a given prescribed amount of fill. For the MDFC PIL-Fill problem, we describe three practical solution approaches based on Integer Linear Programming (ILP-I and ILP-II) and the Greedy method. For the MSFC PIL-Fill problem, we describe an iterated greedy method that integrates call to an industry static timing analysis tool. We test our methods on layout testcases obtained from industry. Compared with the normal fill method [3], our ILP-II method for MDFC PIL-Fill problem 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; and our iterated greedy method for MSFC PIL-Fill problem also shows significant advantage with respect to the minimum slack of nets on post-fill layout.

Categories and Subject Descriptors: B.7.2 [Hardware]: IC; J.6 [Computer Applications]: CAD; F.2.2[Analysis of Algorithms]: Problem Complexity

General Terms: Algorithms, Design, Reliability, Theory

Keywords:VLSI Manufacturability, Dummy Fill Problem, Coupling Capacitance Extraction, Signal Delay, Linear Programming, Greedy Method

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 [3, 4]: 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*. However, the issues associated with capacitance and area fill are complex and there is no existing published work on performance-driven area fill synthesis.[1] Our present work assumes 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 float-

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

[1] Although this concept has been recently mentioned in some startup web sites [10, 13, 14], no details of functionality are given. Currently, metrological methodologies are used to determine the "best" choice of buffer distance, dummy fill type (grounded versus floating), and dummy fill pattern.

DAC 2003, June 2–6, 2003, Anaheim, California, USA.

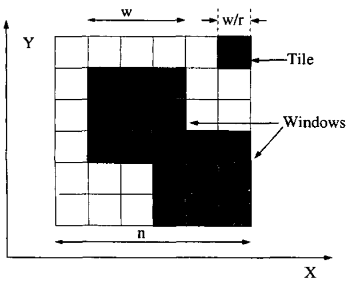

Figure 1: In the fixed r-dissection framework, the n-by-n layout is partitioned by r^2 (here, $r = 3$) distinct overlapping dissections with window size $w \times w$. This induces $\frac{nr}{w} \times \frac{nr}{w}$ *tiles*. Each dark-bordered $w \times w$ window consists of r^2 tiles.

ing fill. Section 4 formulates the PIL-Fill problem with two different objectives, and solution approaches are given in Section 6 and Section 7. Section 8 gives experimental results and we conclude in Section 9.

2. RELATED WORK

According to Stine et al. [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., capacitance minimization does not comprehend the delay and timing slack impact of added capacitance. While no work has (in our opinion) yet addressed the PIL-Fill problem, two related works are of interest.

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. When interconnect lines are more sparsely situated, floating fill has greater performance impact.

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 found by modeling the effects on interconnect capacitance of different design rules (which are consistent with the fill pattern density requirement).

3. CAPACITANCE AND DELAY MODELS

Works on multi-layer interconnect capacitance extraction include 1-D, 2-D, 2.5-D and 3-D analytic models [1, 2, 5, 6, 12]. In general, the capacitance of interest at any node consists of three components: (i) *overlap (area) capacitance*; (ii) *lateral coupling capacitance*. and (iii) *fringe capacitance*; 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 empty sites to satisfy the density requirements. 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

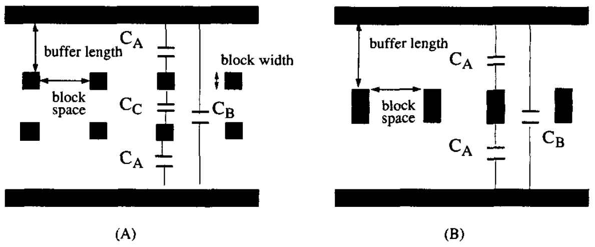

Figure 2: Example configurations of floating dummy fill.

$$C_{orig} = C_B \cdot l = \frac{\varepsilon_0 \varepsilon_r a}{d} \cdot l \qquad (1)$$

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

$$C_{fill} = (\frac{1}{1/C_A + 1/C_C + 1/C_A}) \cdot w \cdot k + C_B \cdot (l - w \cdot k) \qquad (2)$$

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. 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 separated by a column of m dummy features:

$$\begin{aligned} C_{A'} = f(m) &= \frac{\varepsilon_0 \cdot \varepsilon_r \cdot a}{d - m \cdot w} && (3) \\ &\approx C_B + \frac{\varepsilon_0 \cdot \varepsilon_r \cdot a \cdot m \cdot w}{d^2} && (4) \end{aligned}$$

When $w \ll d$, we can further simplify the calculation as a linear one (see Equation (4)), where $\frac{\varepsilon_0 \cdot \varepsilon_r \cdot a \cdot m \cdot w}{d^2}$ is the incremental capacitance due to dummy feature insertion. Then, the total capacitance between two active lines can be estimated as

$$C_{fill} = C_{A'} \cdot w \cdot k + C_B \cdot (l - w \cdot k). \qquad (5)$$

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 is

$$\tau_N = \sum_{i=1}^{N} R_i \sum_{j=i}^{N} C_j = \sum_{i=1}^{N} C_i \sum_{j=1}^{i} R_j \qquad (6)$$

Each node j on the chain contributes to τ_N, 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

$$\Delta\tau_k = \Delta C_i \sum_{j=1}^{i} R_j \qquad (7)$$

Equation (6) implies that Elmore delay is an additive with respect to capacitance along any source-sink path. That is, if we add the coupling capacitance C_x at position x, the delays at all nodes below the position x will increase by $C_x \cdot R_x$. Here, R_x is a constant, and equal to the total resistance between the source and the position x (we will refer to this as *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 two performance-impact limited area fill problem formulations (PIL-Fill) in which the objectives are to minimize performance impact, subject to a constraint of prescribed amounts of fill in every tile.

4.1 Min-Delay-Fill-Constrained Objective

Our *minimum delay with fill constraint*, or MDFC, formulation[2], 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 in the fixed-dissection layout can be considered independently, 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 T such that total impact on delay is minimized.*

For this per-tile MDFC PIL-Fill problem, we use the above capacitance approximations (essentially the same as those in [11]) and the Elmore delay model. Under 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

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

which allows us to directly minimize total sink delay impact over all nets in a given tile.[3]

4.2 Max-MinSlack-Fill-Constrained Objective

A weakness of the MDFC PIL-Fill formulation is that we minimize the total delay impact *independently* in each tile. That is, the impact due to fill features on the signal delay of complete timing paths is not directly considered. Thus, we also propose to maximize the minimum slack of all nets, still subject to a constraint of prescribed amounts of fill in every tile region of the layout. We call this a *maximum min-slack with fill constraint*, or MSFC, formulation.

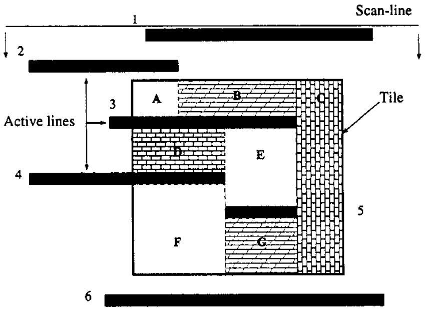

Figure 3: Illustration of scan-line and slack blocks within tile.

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 minimum slack over all nets in the layout is maximized.*

We use a commercial Static Timing Analysis tool (Cadence Pearl) to extract slacks at all pins of each net in the layout.

5. GEOMETRY COMPUTATION

The key computational geometry task in solving PIL-Fill problems is to find all pairs of parallel active line segments, as well as the slack space (i.e., empty sites where fill geometries can be inserted) between each such pair. Without loss of generality, we assume that the routing direction is horizontal on the selected layer.

We define a *slack site column* as a column of available sites for fill features between two active lines or between an active line and a layout boundary. A *slack block* is a maximal contiguous set of slack site columns having equal height (and, due to the fill site grid, equal width). Figure 3 shows seven such slack blocks in a tile. As an example, the fill features located in the slack block C in Figure 3 will affect the coupling capacitance on active lines 1 and 6. We also define the *size of a slack site column* as the number of empty sites in the column available for fill insertion.

To find such slack columns in the layout, we first obtain the position of each active line. After sorting the active lines 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.

6. APPROACHES FOR MDFC PIL-FILL

6.1 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 (4). Without loss of generality, we assume the routing direction on the layer is horizontal, and we also ignore any wrong-direction routing. The MDFC 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;

[2] 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.

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

- $Cap_k \equiv$ incremental capacitance caused by the m_k dummy features in column C_k, calculated according to Equation (4);
- $\Delta\tau_l \equiv$ 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:

$$\sum_{l=1}^{L} W_l \cdot \Delta\tau_l \quad \text{over all active lines} \tag{8}$$

Subject to:

$$F = \sum m_k \quad \text{over all slack columns in tile} \tag{9}$$

$$Cap_k = \frac{\varepsilon_0 \cdot \varepsilon_r \cdot a \cdot w}{{d_k}^2} \cdot m_k \quad \text{for each slack column} \tag{10}$$

$$\Delta\tau_l = \sum_k Cap_k \cdot (R_l + \sum_{s=p}^{k} r_l) \quad \text{for each active line} \tag{11}$$

$$\text{Integer:} \quad 0 \le m_k \le C_k \tag{12}$$

- The objective function (8) 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 (9) ensure that the total number of *covered* (i.e. used) slack sites is equal to the number of dummy features.
- Constraints (10) 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 per slack column, d_k is the distance between them, and w is the dummy feature width.
- Equations (11) 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 leftmost point of the active line in the tile; k is overloaded to also denote the x-coordinate of slack column k.
- Constraints (12) ensure that the number of covered slack sites in any column is no greater than the column size (capacity).

6.2 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 (3) are constant for each pair of active lines, we can pre-build a lookup table $f(n,d)$ that gives the capacitance increment for inserting n fill features between any pair of active lines that are 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:

$$m_{k_n} = \begin{cases} 1 & if\, m_k = n \\ 0 & otherwise \end{cases} \tag{13}$$

Minimize:

$$\sum_{l=1}^{L} W_l \cdot \Delta\tau_l \quad \text{over all active lines} \tag{14}$$

Subject to:

$$F = \sum m_k \quad \text{over all slack columns} \tag{15}$$

$$m_k = \sum_{n=1}^{C_k} n \cdot m_{k_n} \quad \text{for each slack column} \tag{16}$$

$$\sum_{n=1}^{C_k} m_{k_n} = 1 \quad \text{for each slack column} \tag{17}$$

$$Cap_k = \sum_{n=1}^{C_k} f(n, d_k) \cdot m_{k_n} \quad \text{for each slack column} \tag{18}$$

$$\Delta\tau_l = \sum_k Cap_k \cdot (R_l + \sum_{s=p}^{k} r_l) \quad \text{for each active line} \tag{19}$$

$$\text{Integer:} \quad 0 \le m_k \le C_k \tag{20}$$

$$\text{Binary:} \quad m_{k_n} \tag{21}$$

- Constraints (16), (17), (18), and (21) replace constraints (10) in the ILP-I formulation.
- Constraints (16 and 17) imply that m_k can only be assigned one value from 1 to C_k.
- Constraints (18) is the equation for coupling capacitance based on the lookup table. Here, $f(n, d_k)$ is the constant value from the pre-built lookup table.

6.3 Greedy Method

From Equation (11), the impact on delay due to the dummy features depends on the total resistance between the source and the current node. Our final algorithmic approach for the MDFC PIL-Fill problem is to greedily insert dummy features along active line segments where the incremental delay is minimum. This greedy approach is described in Figure 4.[4]

7. ITERATED APPROACHES FOR MSFC PIL-FILL

To maximize minimum slack over all nets in the post-fill layout, we propose an Iterated Greedy approach based on iterations between the static timing analysis (STA) tool and the area fill synthesis. Performance impact due to fill feature insertion during area fill synthesis is written in Reduced Standard Parasitic Format (RSPF) as a file input to STA tool.

This approach uses the same capacitance and delay models as in the MDFC PIL-Fill approaches. After obtaining the density requirements from normal area fill synthesis and slack site columns from the scan-line algorithm, we run the industry STA tool to get the slack values of all input pins in the layout and set the slack of each active line as the minimum slack of its downstream input pins. We consider the slack value of a given slack site column to be the minimum slack of its neighboring active lines. Then, all slack site columns are sorted according to their slack values. Among them, the slack site column with maximum slack value is chosen for fill feature insertion. For each tile intersecting with this slack site column, the number of fill features actually inserted in the column is dependent on the number of required fill features of the tile, the overlapping size of the slack site column, and the column's slack value. Once a feasible number of fill features has been inserted into

[4] 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.

```
Greedy MDFC PIL-Fill Algorithm
Input: the design-rule 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 total delay increase while satisfying
       density requirements
 1. Partition the layout into tiles and sites
 2. Run LP/Monte-Carlo [3] to get the number of required fillFeatures
    ($RF_{ij}$) 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 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.       Get cumulative resistance $\hat{r}_k$ at position $k$ on neighboring active
          lines $l$ and $l'$ as: $W_l(R_l(i,j)+\Sigma_{s=p}^{k} r_l)+W_l'(R_l'(i,j)+\Sigma_{s=p'}^{k} r_{l'})$
12.       Calculate induced coupling capacitances $\hat{Cap}_k$ of column $k$
          as in Equation (3) with $C_k$ dummy features
13.     Sort all slack columns in the tile according to their corresponding
        delay increments as $\hat{r}_k \cdot \hat{Cap}_k$
14.     Initialize the number of filled features for tile $T_{ij}$: $FF_{ij} = 0$
15.     While $FF_{ij} < RF_{ij}$ Do
16.       Select slack column $C_k$ with the minimum corresponding delay
17.       Insert $\min((RF_{ij} - FF_{ij}), C_k)$ dummy features
         in the slack column
18.       Delete the slack column
19.        $FF_{ij} += \min((RF_{ij} - FF_{ij}), C_k)$
```

Figure 4: Greedy MDFC PIL-Fill algorithm.

the tile, the number of required fill features of the tile and the size of the affected slack site column are updated. The added delay is estimated based on our capacitance and delay models, and the slack value of the slack site column updated accordingly. These steps are repeated until fill requirements for all tiles in the layout are met.

To prevent the greedy method from quickly reaching a local minimum, we introduce two variables that enable iterations between STA and area fill synthesis.

- LB_{slack} gives a lower bound on the slack value of slack site columns. Once the largest slack value of any slack site column is less than LB_{slack}, the filling loop is stopped and a new iteration between STA and area fill synthesis is initiated with smaller LB_{slack}.
- UB_{delay} gives an upper bound on the total added delay in the layout. Once the newly added delay during an iteration exceed UB_{delay}, the filling loop is stopped and a new iteration between STA and area fill synthesis is initiated.

Our algorithm is described in detail in Figure 5, where the following definitions are used.

- $RF \equiv$ total number of required fill features in the given layout.
- $RF_{ij} \equiv$ number of required fill features for tile T_{ij}.
- $D_{add} \equiv$ total added delay during the current iteration.
- $S_k \equiv$ slack value of the slack site column k, which is the minimum slack value of its neighboring active lines.
- $S_{max} \equiv$ maximum slack value over all slack site columns.
- $SF_k \equiv$ maximum number of fill features that can be inserted in slack site column k such that the post-fill slack value of the column is still larger than LB_{slack}.
- $C_{k,ij} \equiv$ overlapping size of column k in tile T_{ij}.
- $F_{k,ij} \equiv$ number of inserted fill features in column k in tile T_{ij}.

8. COMPUTATIONAL EXPERIENCE

We have tested our proposed algorithms using five layout test cases, denoted T1, T2, T3, T4 and T5, obtained from industry sources. Each of the test cases was obtained in LEF/DEF format. Signal delay calculation is based on extracted Reduced Standard Parasitic Format (RSPF) files, and "Normal" fill was synthesized using the normal fill method [3] according to the parameters shown in the leftmost column of Table 1.[5]

```
Greedy MSFC PIL-Fill Algorithm
Input: the design-rule correct layout; window size $w$; dissection value $r$;
       the fill pattern (size of fill feature $s$, gap between fill features $g$,
       buffer distance from interconnect $b$), slack lower bound $LB_{slack}$,
       and upper bound of per-iteration incremental delay $UB_{delay}$
Output: filled layout maximizing the minimum slack of all nets while
       satisfying density requirements
 1. Partition the layout into tiles and sites
 2. Run LP/Monte-Carlo [3] to get the number of required fillFeatures
    ($RF_{ij}$) for each tile $T_{ij}$ and total number of required fillFeatures $RF$
 3. Get slack site columns by scanning the layout
 4. Run STA tool with RSPF file to get slacks for all input pins
 5. Calculate the slack of each active line $l$
 6. Calculate slack value $S_k$ for each slack site column
 7. Sort all slack site columns according to their slack values
 8. While ( $RF < 0$ ) Do
 9.     Choose the slack site column $k$ with the maximum slack value $S_{max}$
10.     If ( $S_{max} < LB_{slack}$ )
11.         Update RSPF file with the capacitance increase
12.         Decrease $LB_{slack}$ by given value, Goto step (4)
13.     Calculate $SF_k$ for column $k$
14.     For each tile $T_{ij}$ intersecting with column $k$ Do
15.         Calculate the overlapping size $C_{k,ij}$ of the column in tile $T_{ij}$
16.         Number of fill features to be inserted: $F_{k,ij} = \min(RF_{ij}, C_{k,ij}, SF_k)$
17.         if ( $F_{k,ij} > 0$ )
18.             Fill up the column with $F_{k,ij}$ fill features
19.             Calculate the added delay $d$ due to the $F_{k,ij}$ fill features and
                update the neighboring active lines' delay
20.             $RF_{ij}$ -= $F_{k,ij}$, $RF$ -= $F_{k,ij}$, $SF_k$ -= $F_{k,ij}$, $D_{add}$ += $d$;
21.             if ( $D_{add} > UB_{delay}$ )
22.                 $D_{add} = 0$
23.                 Update RSPF file with the capacitance increase
24.                 Goto step (4)
25. Update RSPF file with the capacitance increase
26. Run STA with RSPF file to check the result
```

Figure 5: Greedy MSFC PIL-Fill algorithm.

8.1 MDFC PIL-Fill Experiments

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 MDFC 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:* Lookup Table Based Integer Linear Programming method; *Greedy:* Greedy method; τ: total delay increase (ns); *CPU:* runtime (seconds).

Table 1 reports the total delay increase over all wire segments due to the "normal" fill method [3], and due to our three performance-

[5] 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.

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 runtime 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 apparently suffers from excessive loss of accuracy. 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.

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 MDFC PIL-Fill synthesis.

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.

8.2 MSFC PIL-Fill Experiments

Testcase	Orig Layout			DenConstr	Normal	MSFC-PIL
T/W/r/	MaxDen	MinDen	minSlack	MinDen	minSlack	minSlack
T3/40/2	0.382	0.086	599.39	0.258	-57.43	369.09
T3/80/2	0.350	0.088	599.39	0.221	-712.06	470.36
T4/40/2	0.341	0.033	1061.56	0.298	-267.70	930.57
T4/80/2	0.325	0.101	1061.56	0.296	9.97	918.28
T5/40/2	0.381	0.091	1974.78	0.264	-582.76	1431.51
T5/80/2	0.357	0.092	1974.78	0.223	-404.16	1846.80

Table 3: Iterated approaches for MSFC PIL-Fill. Notation: *MaxDen:* maximum window density on layout; *MinDen:* minimum window density on layout; *DenConstr:* density requirement specified as a minimum post-fill window density; *MSFC-PIL:* results of MSFC PIL-Fill method; *minSlack:* minimum slack over all nets (ps).

In Table 3, we compare the minimum slack of all nets after the "normal" fill method and after our performance-impact limited fill method, where the density requirement is specified as a post-fill minimum window density. Our experiments show that the fill results from the "normal" fill method may be unacceptable with respect to the minimum slack of nets since these slack values become close to 0 or negative. In contrast, our iterated greedy method for MSFC PIL-Fill performs much better and all post-fill minimum slack values are much larger than 0. The differences between the minimum slack values of "normal" fill result and MSFC PIL-Fill result are show substantial advantages of our approach.

9. CONCLUSIONS AND FUTURE RESEARCH

In this paper, we have developed approximations for the capacitance impact of area fill insertion, and given the first formulations for the Performance Impact Limited Fill (PIL-Fill) problem. We have presented two Integer Linear Programming based approaches and a Greedy method for the MDFC PIL-Fill problem, as well as an iterated greedy method for the MSFC PIL-Fill problem. Experiments on industry layouts indicate that our PIL-Fill methods can reduce the total delay impact of fill, or the impact on minimum slack, by very significant percentages.

Our ongoing research is focused on budgeting slacks along segments so that computationally expensive iteration with STA can be avoided in the optimization procedure. Other research addresses alternative PIL-Fill formulations, e.g., wherein an upper bound on timing impact constrains the minimization of layout density variation.

10. 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*, 1999, pp. 576-580.

[13] *UbiTech. Inc.*, `http://www.ubitechnology.com/`

[14] *XYALIS*, `http://www.xyalis.com/`