Physical Design Techniques for Optimizing RTA-induced Variations

Yaoguang Wei[1], Jiang Hu[2], Frank Liu[3], Sachin S. Sapatnekar[1]
[1]Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN, USA
[2]Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX, USA
[3]IBM Austin Research Lab, Austin, TX, USA
Emails: {weiyg, sachin}@umn.edu; jianghu@ece.tamu.edu; frankliu@us.ibm.com

***Abstract*— At 65nm and below, Rapid Thermal Annealing (RTA) makes a significant contribution to manufacturing process variations, degrading the parametric yield. RTA-induced variability strongly depends on circuit layout patterns, particularly the distribution of the density of the Shallow Trench Isolation (STI) regions. In this work, we investigate a two-step approach to reduce the impact of RTA-induced variations. We first solve a floorplanning problem that aims to reduce the RTA variations by evening out the STI density distribution. Next, we insert dummy polysilicon fills to further improve the uniformity of the STI density. Experimental results show that our floorplanner can reduce the global RTA variations by 39% and the local variations by 29% on average with low overhead compared to a traditional floorplanner, and the proposed dummy fill algorithm can further reduce the RTA variations to negligible amounts. Moreover, when inserting dummy fills, for the layouts obtained by our floorplanner, on average, 24% fewer dummy polysilicon fills are inserted, as compared to the results from a traditional floorplanner.**

I. Introduction

Variations in the semiconductor manufacturing process can cause significant yield loss and performance degradation, and tremendous effort is being expended in addressing these issues. Several manufacturing steps, such as chemical-mechanical polishing, have received substantial attention from the design community. A less well-studied, but increasingly important, issue is related to the effects of thermal annealing, a major contributor to on-chip variability [3].

Rapid thermal annealing (RTA) is widely used during manufacturing [4], and a critical point at which it impacts chip performance is when it is employed in contemporary ultra-shallow junction (USJ) technologies to activate the dopants after implantation. RTA commonly involves the application of high temperature for a short period to perform dopant activation, while limiting unwanted dopant diffusion. Annealing largely determines the concentration of dopant and the junction depth, which in turn determines the threshold voltage, V_T, and also the sheet resistance of the doped regions such as the source, drain and the polysilicon gate. Since these parameters affect the performance of the circuit, it is critically important to ensure that annealing occurs uniformly across the entire die.

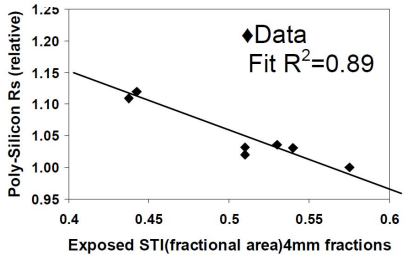


Fig. 1. The variation in the polysilicon sheet resistance R_s correlates well with the exposed STI density averaged over 4mm [3].

The mechanism for RTA involves heating the top side of the wafer, where the active MOSFET devices are located, by exposure to an array of lamps that are pulsed to rapidly transfer radiative heat to the wafer surface. One particular aspect of radiation is that the reflectivity of the surface plays an important role on the amount of heat transferred. The principle is similar to the idea that on a sunny day, shiny or light-colored clothes reflect more light than dark clothes that tend to absorb heat. Across a die, due to differences in the reflectivities of materials and varying pattern densities, different locations will absorb different amount of heat, causing variations in the annealing temperature. This results in within-die and die-to-die variations in the circuit characteristics after RTA.

The resulting variability, often referred to as the pattern effect, has been studied in [3]. Their work investigates the impact of RTA on intra-die variations in the performance, subthreshold leakage, and other parameters, and demonstrates that most of the observed variations, including inverter delay changes, can be accounted for through RTA-driven variations in the FET extrinsic resistance, R_{EXT}, and V_T. Moreover, it shows that the variations correlate with the calculated reflectivity for the lamp RTA spectrum, and hence depend on the mm-scale pattern density of the exposed shallow trench isolation (STI), i.e., the STI not covered by the gate polysilicon. Fig. 1 illustrates how the variation in the polysilicon sheet resistance, R_s, correlates with the exposed STI density. These observations have been qualitatively confirmed by other groups (e.g., in [7]). The work in [14] performs thermal simulation and uses compact models to analyze within-die variability due to pattern-dependent RTA process. Their simulation shows about 20% variation in the leakage and 3% variation in the frequency due to the non-uniformity of the pattern density in the layout. Therefore, it is necessary to optimize RTA-induced variations on a die.

A primary mechanism for reducing the large RTA variations is to achieve an even distribution of exposed STI throughout the layout. There are two ways to achieve this:

- *By rearranging blocks in the layout*: In system-on-chip designs, different circuit modules may have different STI densities, and the differences are especially significant if there are many memory and analog IP blocks. Variations in STI density can be inferred from Fig. 15 in [7] and Fig. 5 in [10]. Physical design plays an important role in achieving STI uniformity, and therefore, floorplanning or placement may be used to alter the STI density distribution to achieve uniform reflectivity.
- *By inserting dummy polysilicon fills*: As demonstrated in [7], [10], RTA variations can be mitigated if dummy fill structures are incorporated to improve the uniformity of the exposed STI. In [10], the dummy fill structure consists of dummy active area and dummy polysilicon gates, while in [7], the dummy fill structures are mainly dummy polysilicon. We assume the latter in our work. In particular, we attempt to minimize the inserted fills to minimize their side-effects, such as unwanted parasitics.

This work addresses both of these aspects. We first propose and formulate the RTA-variation-driven floorplanning problem[1]. We present a floorplanner that optimizes the uniformity of the STI density, in addition to optimizing conventional objectives such as wirelength and area. We emphasize that the contribution of this work is not intended to be in proposing a novel algorithm for floorplanning or dummy fill, but to present a methodology, to develop appropriate models, and to leverage existing techniques wherever possible, to solve the problem of reducing RTA-induced variations.

A significant issue to be addressed is whether RTA optimization should be addressed at the floorplanning level, or at more detailed levels. While methods at all levels of abstraction are appropriate, the greatest amount of flexibility is available at the floorplanning stage, albeit with the least detailed information. This is the classic early planning problem that is faced and surmounted in other aspects of physical design, and our solution is based on several facts. First,

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

[1]Although it is possible and desirable to extend this work to RTA-driven placement, we focus on floorplanning in this work as a first step to RTA-driven physical design.

while the precise STI density of each block may not be known, most design houses perform extensive design reuse, or use similar design styles for similar blocks. Therefore, estimates of block STI densities are realistically available from historical data; in fact, past design data are already often used to predict Rent's parameters or interconnect density. Second, early stage planning for STI density can be made effective throughout the design flow, by propagating the STI densities computed at this early planning stage as constraints at more detailed stages of design. Third, the accuracy and fidelity of the STI model that we use here are appropriate to this level of abstraction: more detailed analysis at finer levels of abstraction may require more detailed heat conduction models. Our experimental results confirm that the amount of improvement available at the floorplanning level is very large (on average, 39% in global variations and 29% in local variations).

In addition to floorplanning optimizations, we present the problem formulation of inserting dummy polysilicon fills that reduce the exposed STI area, thereby helping further improve the RTA variations by ensuring STI uniformity. We show that a linear programming (LP) method, similar to that in [13], can be used to solve the dummy polysilicon filling problem. Our experimental results demonstrate that the proposed dummy fill algorithm can reduce the RTA variations to negligible amounts. We also demonstrate that our two-step approach can significantly reduce RTA variations with very limited impact on conventional design objectives.

II. Background

A. Rapid Thermal Annealing

During ion implantation, nuclear collisions during high energy implantation cause substrate atoms to be displaced, leading to materials defects. RTA is a subsequent annealing process, carried out at high temperature for a short duration, that is used to solve this problem. During RTA, the wafer is heated to around 1000°C for a brief period and then cooled. The RTA process thermally vibrates the atoms, reforms the bonds among them, and activates the dopants [9].

The basic mechanism of RTA is to use radiation to rapidly transfer a large heat flux to the wafer surface; this heat then spreads around the silicon wafer by conduction. Therefore, the surface temperature is due to both radiation and conduction. However, in today's RTA process, the time duration of heating is so short (less than 1 second [11]) that complete thermal equilibrium between conduction and radiation cannot be achieved, and the surface temperature is primarily determined by the ability of different regions of the wafer to absorb heat from the RTA lamps.

As pointed out in [3], the RTA temperature of a point in the layout is highly dependent on the average exposed STI density in the neighborhood of the point. The exposed STI density is averaged over an $l \times l$ region, where l is the heat diffusion length. The parameter, l, is a descriptor of the region over which thermal equilibrium can be reached for the heat diffusion interaction time t, and is given by [3], [5]:

$$l = \sqrt{\frac{\sigma}{c_v \rho} t} \tag{1}$$

where σ, c_v and ρ are the thermal conductivity, specific heat, and density of silicon, respectively. The value of l for the process used in [3] was found to be 4mm.

The reason why the exposed STI density is the most important factor in determining the RTA variability effects is that the reflectivity coefficients of all regions of the wafer are roughly similar, but that of STI is substantially lower [3]. Therefore, STI regions absorb significantly more heat, and transistors that lie in the neighborhood of large STI density regions tend to have lower V_T, because of higher local annealing temperature. Moreover, R_{EXT} in these regions is decreased due to the combination of improved dopant activation and increased gate overlap of the source and drain [11]. The impact of these effects on circuit-level performance is that the delay and leakage of these regions will differ from other regions with higher reflectivity.

Before proceeding further, we will define several concepts related to the STI density in a region. Essentially, the STI density is the ratio of the area of the exposed STI layer (that is not covered by polysilicon) to the total area of the region.

Definition 1 (Local STI density) *The local STI density, $d(x, y)$, at a point (x, y) in the layout is the average STI density defined in a small local square region centered at (x, y). The size of this local region is typically smaller than or in the order of $l/10$ [5], where l is the heat diffusion length, so that the difference of the local anneal temperature in this region is negligible.*

Definition 2 (Effective STI density) *The effective STI density, $e(x, y)$, of a point (x, y) in the layout is the average STI density over a window of size $l \times l$ units, called the* gating window*, centered at (x, y).*

B. Dummy polysilicon filling for RTA

The effective STI density captures the effect of averaging the reflectivity over a region, and the effect of heat transfer by conduction within the wafer, and can be used as a rough predictor of local annealing temperature [11]. To make the annealing temperature uniform, dummy polysilicon fills can be inserted over the STI region to even out the effective STI density, as illustrated in Fig. 2, so that the exposed STI density and the reflectivity of different regions become similar.

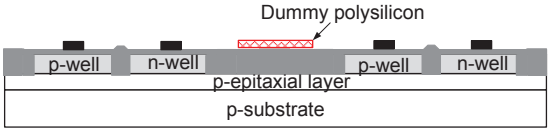


Fig. 2. Cross-section of the wafer, with dummy polysilicon fills inserted before ion implantation and RTA. The dark grey regions are SiO_2 and the dark regions are polysilicon gates.

Dummy polysilicon features are electrically inactive, but can couple capacitively with other lines in neighboring layers. Since these polysilicon lines are left floating, such coupling could cause unpredictable effects on performance, and it is important to minimize the amount of inserted dummy fills.

There are two classes of methods that are used for dummy fill insertion: rule-based and model-based. Analogous to dummy filling methods for oxide chemical mechanical polishing (CMP), rule-based methods work directly on the local STI density, and try to make the local STI density fall within a specified limit after dummy filling. Therefore, in a region with very large STI density, dummy polysilicon fills may be inserted to block off some part of the STI region, in order to bring the local STI density within the desired limit. The problem with rule-based dummy filling is that the limit for the allowable density is typically quite large, so that the prescribed fill density must go through trial-and-error for every design; sometimes, no single value works [13]. Compared to the rule-based approach, model-based dummy polysilicon filling is based on the effective STI density, a more complex metric than the local STI density used in the rule-based method. Model-based methods are analytical and typically more computationally intensive, but are widely accepted to be more effective than rule-based methods.

III. The Effective STI Density

As mentioned before, the effective STI density correlates well with the variations in circuit parameters such as R_s. A key step in our algorithm is the computation of the effective STI density. We first create a formulation for computing the effective STI density, and then describe efficient computational techniques for this purpose.

A. A formulation for the effective STI density

Given the local STI density of each point in the layout $d(x, y)$, we proceed to calculate the effective STI density. By definition, the effective STI density, $e(x, y)$, at a point is the average density in a gating window of size $l \times l$ around the point. Therefore, it can be computed as:

$$e(x, y) = \frac{1}{l \times l} \int_{y-l/2}^{y+l/2} \int_{x-l/2}^{x+l/2} d(u, v)\, du\, dv. \tag{2}$$

Let W and H be the width and height of the floorplan, respectively. Since the die is periodically repeated across the wafer, the RTA environment experiences periodicity: in other words, $d(x, y)$ is periodic.

Therefore, we have

$$d(x + k_1 W, y + k_2 H) = d(x, y), \tag{3}$$

where k_1 and k_2 are two integers such that $(x + k_1 W, y + k_2 H)$ is a valid point on the wafer.

We introduce a gate function $g(x, y)$ defined below:

$$g(x, y) = \begin{cases} \frac{1}{l \times l} & -l/2 \le x, y \le l/2, \\ 0 & \text{otherwise}, \end{cases} \tag{4}$$

where l is the heat diffusion length for a RTA process.

From (2) and (4), it can easily be verified that $e(x, y)$ can be calculated by convolving $d(x, y)$ with $g(x, y)$:

$$e(x, y) = d(x, y) \otimes g(x, y), \tag{5}$$

where $\otimes$ is the circular convolution operator.

Since the effective STI is a long-range effect, and since the calculation of circular convolution in the continuous field involves computationally-intensive symbolic double integration, we choose to compute it over a discrete field. We tessellate the layout into M rows and N columns of tiles, each with side equal to s. The value of s can be chosen so that it is sufficiently small. As mentioned in Definition 1, the difference of the local anneal temperature in a region with size $l/10 \times l/10$ is negligible [5], so we use $s = l/10$ in this work. The heat diffusion length, l, in the continuous space maps on to a length of L tiles in the discrete space.

Let t_{ij} be the tile in the ith row and the jth column of the partitioned layout, $0 \le i \le (M-1), 0 \le j \le (N-1)$. Since s is small enough, the local STI densities of all the points in t_{ij} are assumed to equal that of the central point of t_{ij}, denoted as $d(i, j)$. We refer to this as the local STI density of the tile, and it is computed as the STI density averaged over the area of the tile. Similarly, we discretize the gate function, $g(i, j)$, as the discrete version of $g(x, y)$, and $e(i, j)$ as the discrete counterpart of $e(x, y)$.

We define the discrete gating window for a tile t_{ij} as a region of $L \times L$ tiles, where L is an odd number, centered around t_{ij}. The discretized version of (4) is

$$g(i, j) = \begin{cases} \frac{1}{L \times L} & -\lambda \le i, j \le \lambda, \\ 0 & \text{otherwise}, \end{cases} \tag{6}$$

where $\lambda = (L-1)/2$. The discretized effective STI function, $e(i, j)$ is the discrete analog of (5):

$$e(i, j) = d(i, j) \otimes f(i, j), \tag{7}$$

where $\otimes$ now represents a discrete circular convolution.

B. Efficient computation techniques

We will now discuss three methods for computing $e(i, j)$. The first method uses the definition of circular convolution to calculate $e(i, j)$ in the space domain:

$$\begin{aligned} e(i, j) &= \sum_{i', j' \in G(i,j)} \sum d(i + i', j + j') \times g(i', j') \\ &= \frac{1}{L \times L} \sum_{i', j' \in G(i,j)} \sum d(i + i', j + j'), \end{aligned} \tag{8}$$

where $G(i, j)$ is the gating window of $L \times L$ tiles about (i, j). The runtime required to calculate $e(i, j)\ \forall\ i, j$ is $O(nL^2)$, where $n = M \times N$ is the total number of tiles in the layout.

The second method performs the computation in the frequency domain, using the Fast Fourier Transformation (FFT). We first pad the 2-D sequence $g(i, j)$ to the same size as $d(i, j)$, and then compute $e(i, j)$ as:

$$e(i, j) = \text{IFFT}\left(\text{FFT}(d(i, j)) \times \text{FFT}(g(i, j))\right). \tag{9}$$

where FFT and IFFT are the FFT operator and inverse FFT operator, respectively. The time complexity of this computation is $O(n \log n)$.

Finally, we introduce the third method, which is the procedure used in our algorithm. Considering a special property of the gating function, we find that we can compute the effective STI density of all the tiles in the layout by an incremental method. The basic idea of the method is to reuse prior computations as much as possible.

Let W_{ij} be the set of tiles in the gating window for t_{ij}. We will assume that the origin, $(i, j) = (0, 0)$, is at the bottom left corner of the layout. As compared to $W_{i,j-1}$, the elements of W_{ij} are largely identical, except that the leftmost column of $W_{i,j-1}$ is removed, and the a new column is added to the right. In other words, the computation of $e(i, j)$ can largely reuse the computation for $e(i, j-1)$, except that the contribution from one column must be removed, and that of another one added. An example for this is shown in Figure 3.

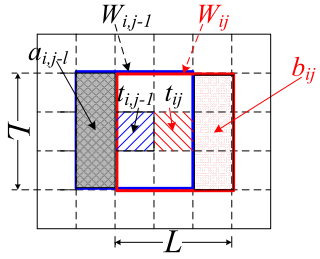


Fig. 3. An example of incrementally computing $e(i, j)$ according to $e(i, j-1)$.

Let $a_{i,j-1}$ be the leftmost column of $W_{i,j-1}$ and b_{ij} be the rightmost column of W_{ij}. Let $M_a(i, j-1) = \sum_{t_{pq} \in a_{i,j-1}} d(p, q)$ and $M_b(i, j) = \sum_{t_{pq} \in b_{ij}} d(p, q)$. Then, it is easily seen that $e(i, j) = e(i, j-1) + (M_b(i, j) - M_a(i, j-1))/L^2$.

The contents of $M_a(i, j-1)$ can be incrementally computed in a similar manner: note that $M_a(i-1, j-1)$ and $M_a(i, j-1)$ have most terms in common, except that the latter adds a new row on top and removes the one at the bottom.

The pseudocode for this incremental method to compute $e(i, j)$ is shown in Algorithm 1. It can be shown that the computational complexity of this method is $O(MN + (M+N)L)$. If $L < \min(M, N)$, as is typically the case since the size of die is usually larger than L, the runtime is $O(n)$.

Algorithm 1 The incremental method for computing the effective STI densities for all tiles in the layout. In the algorithm, v_r is a vector, $\lambda = (L-1)/2$, M_c is a $M \times N$ matrix, and $\psi(i, j)$ is a function to transform any (i, j) to (i', j'), where $i' = i \bmod M, j' = j \bmod N$.

for $i = 0$ to $M-1$ **do**
  $v_r(i) \leftarrow \sum_{j=-\lambda}^{\lambda} d(\psi(i, j))$
end for
for $j = 0$ to $N-1$ **do**
  $M_c(0, j) \leftarrow \sum_{i=-\lambda}^{\lambda} d(\psi(i, j))$
end for
for $i = 1$ to $M-1$ **do**
  for $j = 0$ to $N-1$ **do**
    $M_c(i, j) \leftarrow M_c(i-1, j) + d(\psi(i+\lambda, j)) - d(\psi(i-\lambda-1, j))$
  end for
end for
$e(0, 0) \leftarrow \sum_{j=-\lambda}^{j=\lambda} M_c(\psi(0, j))$
// Compute $e(i, j)$ for the 0-th column tiles
for $i = 1$ to $M-1$ **do**
  $e(i, 0) \leftarrow e(i-1, 0) + v_r(\psi(i+\lambda)) - v_r(\psi(i-\lambda-1))$
end for
// Compute $e(i, j)$ for the rest of the tiles
for $i = 0$ to $M-1$ **do**
  for $j = 1$ to $N-1$ **do**
    $e(i, j) \leftarrow e(i, j-1) + M_c(\psi(i, j+\lambda)) - M_c(\psi(i, j-\lambda-1))$
  end for
end for
for each $e(i, j)$, $e(i, j) \leftarrow e(i, j)/L^2$

C. Finding the discretized local STI density

In this section, we will show how to compute the local STI density for every tile in the layout: this is used in the computation of effective STI density.

It is well known that the STI density of a floorplan block can vary significantly over the area of the block, unless the block is a very regular structure, such as a memory module. To address the

issue, we allow the specification of different STI densities to different subblocks of a block if its size is larger than $s \times s$, where $s = l/10$, since local annealing temperature variations in an $s \times s$ window are negligible.

Then the discretized local STI density function, $d(i, j)$, can be computed based on the floorplan information, which provides the local STI densities of each subblock (as explained in Section I). For a tile, t_{ij}, that overlaps with one or more subblocks, each with (possibly) different average STI densities, and/or with dead space, the value of $d(i, j)$ should be the STI density averaged over the area of t_{ij}. Dead space on the chip is assumed to have a user-specified density of η; typically, $\eta = 1$.

IV. RTA-driven floorplanning

The RTA-variation-driven floorplanning problem is stated as follows:

Given a set of circuit blocks, along with their widths, heights, pin positions, and local STI densities of all the subblocks, and a netlist that captures the connections between the blocks, find a legal floorplan such that a weighted cost function of area, wirelength and effective STI density uniformity is minimized.

Our floorplanner is an adaptation of the Parquet package [1], [2], which is based on simulated annealing (SA) and the sequence pair representation [8]. The FAST-SP algorithm [12] is used to evaluate a sequence pair within the algorithm. We suggest new objective functions to drive simulated annealing, introduced in Section IV-A. Moreover, we use a heuristic to improve the performance of the algorithm, as described in Section IV-B.

A. Cost function

Our goal is to optimize the uniformity of the distribution of the effective STI density. We use the weighted sum

$$\mathcal{U} = \mathcal{R} + \gamma \mathcal{G} \tag{10}$$

as the metric to represent this uniformity, where $\mathcal{R}$ is the global variation, i.e., the difference between the maximum and the minimum, of the effective STI density in the floorplan, $\mathcal{G}$ is the sum of the absolute values of the gradients of effective STI density, and γ is a user-defined parameter. This first component, $\mathcal{R}$, is directed towards reducing global variations of the effective STI density, while $\mathcal{G}$ serves to reduce local variations. The utility of the latter is, for example, in reducing the variation in the clock skew between electrically adjacent registers which are also (usually) spatially adjacent. However, minimizing $\mathcal{G}$ does not obviate the need for the metric $\mathcal{R}$, since even a small level of local variation can accumulate into a large undesirable global variation.

To further define $\mathcal{G}$, we define the horizontal gradient, h_{ij}, and vertical gradient, v_{ij}, of the effective STI density of a tile t_{ij} as:

$$h_{ij} = \begin{cases} 0 & \text{if } j = N - 1, \\ e(i, j+1) - e(i, j) & \text{otherwise,} \end{cases} \tag{11}$$

$$v_{ij} = \begin{cases} 0 & \text{if } i = M - 1, \\ e(i+1, j) - e(i, j) & \text{otherwise.} \end{cases} \tag{12}$$

We can then compute $\mathcal{G}$ as:

$$\mathcal{G} = \sum_{\text{all } t_{ij}} (|h_{ij}| + |v_{ij}|). \tag{13}$$

In practice, a larger weight should be assigned to $\mathcal{G}$, which means that γ should be larger than 1, because the local variation between two adjacent tiles tends to have a larger impact on circuit design compared with the variation between two tiles far from each other.

However, in optimizing STI uniformity through floorplanning, it is important to also factor in conventional floorplanning metrics. The following cost function is used in the floorplanner:

$$\mathcal{C} = \overline{\mathcal{A}} + \alpha \overline{\mathcal{W}} + \beta \left(\overline{\mathcal{R}} + \gamma \overline{\mathcal{G}} \right), \tag{14}$$

where $\mathcal{C}$ is the total cost; $\overline{\mathcal{A}}$ is the normalized area of the floorplan; $\overline{\mathcal{W}}$ is the normalized total wirelength of the floorplan; $\overline{\mathcal{R}}$ and $\overline{\mathcal{G}}$ are the normalized versions of $\mathcal{R}$ and $\mathcal{G}$ from (10), respectively, and serve as metrics of the uniformity of effective STI density in the floorplan; and α, β and γ are user-defined parameters. Note that during SA, we compute the change in the cost, Δ, which is used in the Metropolis rule, using the relative change of each component:

$$\Delta = \frac{\Delta \mathcal{A}}{\mathcal{A}_{blocks}} + \alpha \frac{\Delta \mathcal{W}}{\mathcal{W}_{old}} + \beta \frac{\Delta \mathcal{R}}{\mathcal{R}_{old}} + \beta\gamma \frac{\Delta \mathcal{G}}{\mathcal{G}_{old}}. \tag{15}$$

Here, $\Delta\mathcal{A}$, $\Delta\mathcal{W}$, $\Delta\mathcal{R}$ and $\Delta\mathcal{G}$ are the change in $\mathcal{A}$, $\mathcal{W}$, $\mathcal{R}$ and $\mathcal{G}$, respectively. This is consistent with normalized objective function in (14), with the normalization carried out relative to $\mathcal{A}_{blocks}$, the total area of all the blocks, and $\mathcal{W}_{old}$, $\mathcal{R}_{old}$, and $\mathcal{G}_{old}$, the old values of $\mathcal{W}$, $\mathcal{R}$ and $\mathcal{G}$, respectively. This is also the way used in Parquet.

B. Heuristics

Our implementation is based on the Parquet package [1], [2]. We make the following major modifications: first, we add our proposed algorithms to make Parquet capable of optimizing RTA variations; second, when optimizing RTA variations, we change the algorithm to two-stage simulated annealing.

In the first stage, a conventional SA is performed, and only the area and wirelength are optimized. After the first stage ends, we start the second stage simulated annealing from a low temperature (100 in our algorithm), using the best solution obtained in the first stage as the initial solution. In this stage, the cost function (14) is used to evaluate a floorplan solution. Since the start temperature is low, the second stage serves more like a local search with certain hill-climbing ability.

In order to improve the performance of our algorithm, we introduce an effective heuristic in the second stage to handle the effect of the $\mathcal{U}$ term. A penalty function is introduced for this term in the cost function to add barriers for accepting a floorplan candidate with better STI uniformity but much worse area. We observe that for a given benchmark, a floorplan solution with larger area contains more area of dead space, which is assumed to have a high density, η, of STI and tends to mitigate the variations in the effective STI density of different blocks in the layout. Therefore, a floorplan solution with larger area tends to have a smaller $\mathcal{U}$ term (i.e., the $\mathcal{R}$ and $\mathcal{G}$ terms) in (14). However, in practice, a circuit with large area of dead space is not desirable, and therefore, we apply a penalty function $\theta = \phi_c/\phi_t$ for the $\mathcal{U}$ term, where ϕ_c is the dead space ratio for the current floorplan solution and and ϕ_t is the targeted dead space ratio that the designer wishes to achieve after floorplanning. With this penalty function, when calculating Δ in (15), each term of $\mathcal{R}$, $\mathcal{R}_{old}$, $\mathcal{G}$ and $\mathcal{G}_{old}$ is scaled by its own value of θ. Using this penalty function, when the current floorplan solution has a large area and then a large ϕ_c, a large penalty factor will be assigned to $\mathcal{U}$, which makes this solution less likely to be accepted in SA. In our experiments, $\phi_t = 0.1$.

V. Dummy polysilicon filling

After the floorplan has been optimized, we then insert dummy polysilicon fills to further improve the uniformity of effective STI density. Since the problem structure is somewhat similar to that of oxide CMP fill, our method is based on an adaptation of the oxide CMP fill solution in [13]. The approach in [13] has two steps: global dummy fill assignment using LP, followed by local dummy fill insertion where the details of the fills at the local level are determined. In this work, we focus on the first step, since the methods for detailed local dummy fill insertion are very similar to that in CMP dummy fill, and the reader is referred to [13] for details on issues such as the determination of the shapes and locations of dummy fills.

Next, we present the LP formulation for the dummy polysilicon filling problem. Compared to the method in [13], additional constraints to control the gradient of the effective STI density are introduced.

As before, the layout is partitioned to M rows and N columns of tiles, so that the total number of tiles is given by $n = MN$. In our work, the amount of dummy fill refers to the ratio of the area of the dummy fills in a tile to the total area of the tile. We define x_{ij} to be the amount of dummy fill inserted into tile t_{ij}, x^a_{ij} to be the upper bound of dummy fill that can be inserted into t_{ij}, and d^0_{ij} to be the initial local STI density in t_{ij}. The local STI density of t_{ij} after dummy filling can be calculated as:

$$d(i, j) = d^0(i, j) - x_{ij}. \tag{16}$$

Note that after inserting dummy polysilicon fills, the local STI density is *reduced*, and therefore $d(i,j)$ will be no larger than $d^0(i,j)$. The effective STI density can be calculated using Algorithm 1.

We formulate the problem to ensure that the global variation and the gradients of the effective STI density are within a small range. Let ε_1 be the required bound for the global variation of the effective STI density, and ε_2 be the required bound for the gradient of the effective STI density of any two adjacent tiles. The cost of inserting dummy fills reflects the side-effects of dummy fills, and may be different in different tiles. Therefore, a weight c_{ij} could be assigned to the dummy fills inserted into tile t_{ij}, and this can be calculated using the cost zone method in [13]. The cost of the dummy fills inserted in t_{ij} can then be written as $c_{ij} \cdot x_{ij}$.

To minimize the total cost of the dummy fills, the dummy polysilicon filling problem is formulated as the following LP problem, with $O(n)$ variables and $O(n)$ constraints:

$$\text{minimize} \sum_{\text{all } ij} c_{ij} \cdot x_{ij} \tag{17}$$

subject to:

$$0 \le x_{ij} \le x^a_{ij}, \forall\, i,j, \tag{18}$$

$$e_L \le e(i,j) \le e_H, \forall\, i,j, \tag{19}$$

$$e_H - e_L \le \varepsilon_1, \tag{20}$$

$$|e(i+1,j) - e(i,j)| \le \varepsilon_2,\ 0 \le i \le M-2, \forall\, j, \tag{21}$$

$$|e(i,j+1) - e(i,j)| \le \varepsilon_2,\ 0 \le j \le N-2, \forall\, i, \tag{22}$$

where e_H and e_L are auxiliary variables that determine the maximum and minimum STI density, respectively. In this formulation, the RTA variations are controlled by constraints (19)$\sim$(22). The inputs are d^0_{ij}, x^a_{ij}, c_{ij}, ε_1 and ε_2; the output is x_{ij}.

VI. Experimental results

A. RTA-aware floorplanning

We have implemented and tested the algorithm on a 64-bit Linux machine with a 2.6 GHz AMD® Opteron® 2218 processor and 2GB memory. We call our implementation as pRTA, and we compare its results with those from the original Parquet, with only area and wirelength optimized.

In our experiments, the RTA profile for 65nm technology is used, and we take $l = 4$mm, in accordance with [3]. The side of a tile, s, equals $l/10$. Two groups of standard benchmarks from the MCNC and GSRC suites are used to test the programs, and basic information about the circuits is listed in Table I. Since some of the benchmarks are much smaller than $l \times l$, we scale their size by a ratio, S, as listed in the table. The areas of the scaled circuits vary from about 7mm×7mm to 15mm×15mm.

TABLE I
Benchmark characteristics: here, S is area scaling factor, and A_S is area of blocks, after scaling, in mm^2.

Circuit	apte	xerox	hp	ami33	ami49	n100	n200	n300
# blocks	9	10	11	33	49	100	200	300
A_S	47	77	79	116	142	173	191	230
S	1	2	3	10	2	31	33	29

As mentioned in Section III-C, large blocks are likely to have different local STI densities in different subblocks. Since data on STI densities are not readily available for these benchmarks, we generate these data pseudorandomly in the following way[2]. First, we select the average STI density, g_i, of each block B_i to be a random number between 0.25 and 0.75. Next, we set a variation range, v_i, for the local STI densities of all of the subblocks within a block: in this paper, this value is set to 0.3. Then the local STI density of a subblock on B_i will be chosen randomly between $\max(g_i - v_i, g_{min})$ and $\min(g_i + v_i, g_{max})$, where $g_{min} = 0.15$ and $g_{max} = 0.85$ are, respectively, assumed as the possible minimal and maximal STI density for a logic subblock[3] in practice. Note that these steps are performed after scaling the benchmarks, so that the distribution of STI remains realistic in spite of the artificial scaling operation. All the random numbers generated here are based on a uniform distribution, using a function from the standard Unix C/C++ library. Further, we set the local STI density of the dead space to $\eta = 1$ in our experiments, and assume that the space between the dies on the wafer is negligible.

For a fair comparison, we use all of the default parameters of Parquet (which have been tuned by the authors of the package): using the notation in (14), we choose

$$\alpha = 1.0, \beta = 0, \tag{23}$$

so that the optimization objectives are $\mathcal{A}$ and $\mathcal{W}$. For pRTA, we tune the weights for the circuit ami49, and then use these weights for all the circuits: specifically, we use

$$\alpha = 1.43, \beta = 0.43, \tag{24}$$

so that $\mathcal{A}$, $\mathcal{W}$, $\mathcal{R}$, and $\mathcal{G}$ are all optimized. To place more importance on local RTA variations, we use $\gamma = 10$. For completeness, we will list ami49 in our table of results, but the gains on this circuit can be appropriately discounted since the parameters were tuned on it.

Since Parquet and pRTA are based on a stochastic algorithm, the results from different runs can be different. We run each algorithm for 10 runs, and list the best result over the 10 runs in Table II.

To demonstrate the impact of our approach on the RTA variations directly, we determine the relationship between the polysilicon sheet resistance R_s and the effective STI density in Fig. 1, based on measured data from [3], as:

$$y = -0.9267x + 1.5223, \tag{25}$$

where y denotes R_s in relative units, and x is the effective STI density.

Using (25), from the variation in the effective STI density in a floorplan, we can calculate and report the variations in R_s for the floorplan. Recall that in Section IV-A, we had defined $\mathcal{R}$ and $\mathcal{G}$ as, respectively, the global and local variation in the effective STI density. We define $\mathcal{R}_{R_s}$, the global variation of R_s, as the difference between the maximum and minimum value of R_s over the layout. The local variation, $\mathcal{G}_{R_s}$, of R_s is defined analogously to (11), (12), and (13), with $e(.,.)$ in each equation substituted by $R_s(.,.)$, and represents the sum of the absolute values of the gradients of R_s over all tiles. We also report the maximum gradient, σ_{R_s}, computed as the maximum, over all tiles, of the absolute values of h_{ij} or v_{ij}, corresponding to the gradient of R_s.

A comparison of the results of pRTA and Parquet is shown in Table II, for the benchmark circuits listed in Table I. We show comparisons of the percentage area usage, the total wirelength, the global variation, $\mathcal{R}_{R_s}$, the local variation, $\mathcal{G}_{R_s}$, and the maximum gradient, σ_{R_s}. From the table, we can see that with tolerable overheads in area and wirelength, the proposed algorithm is effective in reducing the variations in the values of circuit parameters affected by RTA variations. Although our results are shown for variations in R_s, variations in R_{EXT} and V_T are similar in nature since they both are shown to have excellent linear correlations with R_s [3]. Compared with a floorplanner only optimizing traditional objectives, on average, our algorithm can reduce global variation in R_s by 39%, local variations by 29%, and maximum gradient by 16.6%, with almost the same area and wirelength. Note that in fact, our floorplanner obtains a little better area usage than Parquet on average, since a *larger* value for area usage is better.

The runtime of our algorithm is acceptable even for the largest circuit in the benchmarks, which is about 1.4 hours for 10 runs. The major reason for the increase of runtime of our algorithm is that the evaluation of the effective STI density for a floorplan solution incurs a significant computational expense, about 8× longer than the evaluation of area and wirelength. This is mainly due to the large number of tiles in a layout and subblocks in a block. For example, circuit ami49 has 49 blocks and 408 nets, but about 1000 tiles and 1000 subblocks. Therefore, though the proposed algorithms to compute the effective STI density have linear runtime complexity, the runtime is noticeably longer compared with the evaluation of the area and wirelength. In spite of this, the runtime is very reasonable in absolute terms.

To present the improvement of our algorithm in the RTA variations more intuitively, the topographies of R_s in the layouts obtained by

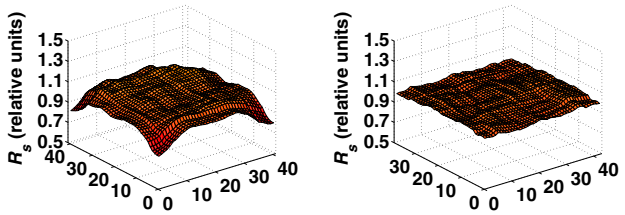
[2] In real designs, the STI density information can be usually obtained, as explained in Section I.

[3] Note that for a tile containing some dead space, its local STI density can be larger than 0.85.

TABLE II
COMPARISON OF PARQUET AND PRTA.

Circuit	Area usage(%)		Wire length (mm)		$\mathcal{R}_{R_s}$		$\mathcal{G}_{R_s}$		σ_{R_s}		Total time(s)	
	Parquet	pRTA	Parquet	pRTA	Parquet	pRTA	Parquet	pRTA	Parquet	pRTA	Parquet	pRTA
apte	96.09	97.90	518.07	519.34	0.15	0.08	6.08	4.32	0.044	0.032	1.2	13.1
xerox	94.11	94.11	749.85	778.20	0.21	0.21	10.69	10.34	0.035	0.044	3.0	26.2
hp	89.14	89.07	652.07	618.76	0.27	0.24	13.69	11.67	0.055	0.043	1.5	28.8
ami33	88.89	92.44	694.65	715.77	0.24	0.18	16.90	13.95	0.049	0.050	8.3	120.4
ami49	88.18	92.52	1409.23	1482.13	0.30	0.20	22.36	16.96	0.054	0.051	22.0	228.4
n100	90.64	90.29	9813.00	10109.40	0.18	0.08	19.69	10.87	0.044	0.039	90.4	682.2
n200	88.39	87.80	18790.10	20083.00	0.21	0.05	21.68	8.31	0.043	0.016	393.6	2312.8
n300	87.90	85.93	19884.80	20414.30	0.30	0.10	28.62	18.27	0.050	0.033	813.7	5120.8
Compare	1.00	1.01	1.00	1.02	1.00	0.61	1.00	0.71	1.000	0.834	1.0	10.4

Parquet and pRTA for circuit n300 are shown in Fig. 4. The x and y axes represent the physical coordinates with one unit equal to 0.4mm, and the z axis shows the R_s value at each location in relative units. From the plots, we can clearly see that our algorithm reduces both the global and local variations in R_s induced by RTA significantly.



(a) (b)

Fig. 4. Topographies of R_s in the layouts obtained by Parquet and pRTA for circuit n300: (a) without RTA optimization (Parquet), and (b) with RTA optimization (pRTA). Note that the STI density is inversely proportional to R_s.

B. Dummy polysilicon filling

In the second step of our optimization framework, we test whether the variations in circuit parameters such as R_s are still larger than the required bounds after floorplanning. If so, we insert dummy polysilicon fills into the layout to further reduce the variations, using the formulation presented in Section V.

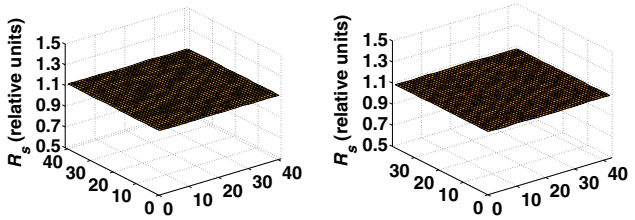
The settings for the dummy fill procedure are as follows. In a tile t_{ij}, we assume that the lower bound of $d(i,j)$ after dummy filling is 0.15, which is reserved to consider the requirements of design rules, e.g., minimal space between dummy fills and polysilicon gates, and to ensure the manufacturability. Therefore $x^a_{ij} = \max(d^0(i,j) - 0.15, 0)$, where max operation is used to ensure x^a_{ij} not negative. For simplicity, the weight c_{ij} for the cost of the dummy fills in each tile is set to 1 in our experiments. The parameters of (20) $\sim$ (22) are chosen as $\varepsilon_1 = 0.0108$ and $\varepsilon_2 = 0.0022$. Based on (25), these values correspond to a global variation of 1% in R_s, and a maximum gradient for R_s of 0.2%. A commercial LP solver ILOG CPLEX® [6] is used to solve the dummy polysilicon filling problem. The hardware used is the same as that used to test floorplanners.

The results are listed in Table III, and they show that to achieve the same requirements of uniformity in the effective STI density, the amount of dummy fills inserted can be reduced by 24%, on average, using the floorplan solutions obtained by pRTA instead of those obtained by Parquet. Therefore, we can see that our floorplanning algorithm has two benefits in improving the RTA variations: first, it provides a floorplan solution with reduced RTA variations directly, which is useful when dummy polysilicon filling is not suitable; second, when inserting dummy polysilicon, fewer dummy fills are inserted, which can reduce the coupling effects induced by dummy fills. Specifically, take the largest circuit n300 in the benchmarks as an example. By using our two-step approach, the global variation in R_s is reduced from 0.30 to 0.01, while the maximum gradient of R_s is reduced from 0.050 to 0.002, with 24% fewer dummy fills inserted. The runtime of the dummy filling method is quite short, and is no more than 16 seconds for all the benchmarks.

The topographies of R_s for circuit n300 after dummy filling are shown in Fig. 5 to demonstrate the effectiveness of the dummy filling algorithm. The x, y and z axes have the same meanings and units as those in Fig. 4. The figures show that after dummy filling, both the global and the local variations are negligible.

TABLE III
DUMMY FILLING RESULTS FOR THE FLOORPLANS OBTAINED BY PARQUET AND PRTA. THE COLUMN "RATIO" DENOTES THE RATIO OF THE TOTAL AREA OF THE DUMMY FILLS TO THE TOTAL STI AREA IN THE LAYOUT.

Circuit	Area of fill(mm^2)		Ratio(%)		Fill time(s)	
	Parquet	pRTA	Parquet	pRTA	Parquet	pRTA
apte	7.29	5.60	24.95	20.40	0.8	0.9
xerox	14.71	12.02	30.05	24.56	2.2	2.2
hp	14.83	12.91	28.80	27.09	1.9	2.4
ami33	21.15	17.89	28.55	24.79	5.6	4.9
ami49	30.66	24.00	33.28	29.08	7.3	4.7
n100	25.18	16.30	23.90	15.35	8.9	9.1
n200	24.52	15.93	19.39	13.02	12.7	16.0
n300	31.84	23.43	20.80	15.37	13.2	13.5
Compare	1.00	0.76	1.00	0.80	1.0	1.0



(a) (b)

Fig. 5. Topographies of R_s in the layouts obtained by Parquet and pRTA for circuit n300 after dummy polysilicon filling: (a) without RTA optimization, and (b) with RTA optimization. Our dummy filling algorithm can produce even profiles for both layouts, but as shown in Table III, the overhead of (a) is significantly larger.

REFERENCES

[1] S. Adya et al. `http://vlsicad.eecs.umich.edu/BK/parquet/`.

[2] S. N. Adya and I. L. Markov. Fixed-outline floorplanning through better local search. In *Proc. ICCD*, pages 328–334, 2001.

[3] I. Ahsan et al. RTA-driven intra-die variations in stage delay, and parametric sensitivities for 65nm technology. In *IEEE Symp. on VLSI Tech.*, pages 170–171, 2006.

[4] S. A. Campbell. *The Science and Engineering of Microelectronic Fabrication*. Oxford University Press, USA, New York, NY, 2nd edition, 2001.

[5] A. Colin et al. Simulation of the sub-melt laser anneal process in 45 CMOS technology–Application to the thermal pattern effects. *Materials Science and Engineering: B*, 154-155:31–34, 2008.

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

[7] K. Kuhn et al. Managing process variation in Intel's 45nm CMOS technology. *Intel Technology Journal*, 12(2):93–109, 2008.

[8] H. Murata et al. Rectangle-packing based module placement. In *Proc. ICCAD*, pages 472–479, 1995.

[9] J. M. Rabaey et al. *Digital Integrated Circuits: A Design Perspective*. Prentice Hall, Upper Saddle River, NJ, 2nd edition, 2003.

[10] J. Scott et al. Reduction of RTA-driven intra-die variation via model-based layout optimization. In *Proc. Symp. VLSI Tech.*, pages 152–153, 2009.

[11] S. K. Springer et al. Modeling of variation in submicrometer CMOS ULSI technologies. *IEEE Trans. Elect. Dev.*, 53(9):2168–2178, 2006.

[12] X. Tang and D. F. Wong. FAST-SP: A fast algorithm for block placement based on sequence pair. In *Proc. ASPDAC*, pages 521–526, 2001.

[13] R. Tian et al. Model-based dummy feature placement for oxide chemical-mechanical polishing manufacturability. *IEEE Trans. on Comput.-Aided Des.*, 20(7):902–910, 2001.

[14] Y. Ye et al. Variability analysis under layout pattern-dependent rapid-thermal annealing process. In *Proc. DAC*, pages 551–556, 2009.