Mask Assignment and DSA Grouping for DSA-MP Hybrid Lithography for sub-7nm Contact/Via Holes

Yasmine Badr, *Student Member, IEEE,* Andres Torres, *Member, IEEE,* Puneet Gupta, *Member, IEEE*

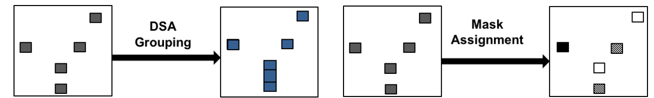
***Abstract*—Directed Self Assembly (DSA) is a very promising candidate for the sub-7nm technology nodes. To print such small dimensions, Multiple Patterning (MP) is likely to be used to print the guiding templates for DSA. Therefore algorithms are required to perform the DSA grouping at the same time as the mask assignment. In this work, we present an optimal Integer Linear Program (ILP) to solve this problem for two schemes of hybrid DSA-MP process. Scalable heuristic algorithms are also proposed to solve the same problem. In comparison to the ILP, the proposed heuristics are 4x-213x faster, and result in an increase of total number of violations by 4%-29%.**

Keywords*—Directed Self Assembly, Multiple Patterning, Design and Technology Co-optimization, sub-7nm

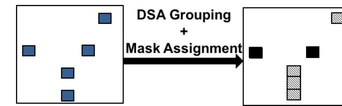
I. INTRODUCTION

In continuous search for new technologies to enable the sub-7nm nodes, Directed Self Assembly (DSA) has presented itself as a strong candidate, especially with the continuous delay of Extreme Ultraviolet Lithography (EUVL). Even if EUVL gets into production, there are far more challenges with the transition to high Numerical Aperture (high-NA) EUVL which will be needed for sub-11nm resolution, making the partnership of EUVL with Multiple Patterning (MP) an alternative option [1]. Thus, Multiple Patterning is expected to enable several sub-7nm nodes. With the cost being the main drawback of MP and with DSA having native frequency multiplication properties, substituting one mask in an MP process with DSA is a tempting cost reduction[2]. In addition, DSA has been reported to possess significant rectification capability in Critical Dimension Uniformity (CDU) and Edge Roughness for contacts[3]. DSA has been successfully demonstrated for contact holes (for e.g., [4]) and lamellae (for e.g., [5]). Since DSA is capable of printing dense nano features of roughly uniform dimensions [6], it is a very good fit for contact and via layers.

In this work, we focus on the hybrid DSA-MP process for contact/via holes and study the problem of simultaneous MP decomposition and DSA grouping. DSA Grouping is the task of assigning contacts into groups such that each group is self assembled in the same guiding template (Figure 1a), while MP decomposition is determining the mask for every polygon (Figure 1b). The guiding templates for DSA are assumed to be printed using 193nm Immersion Lithography (193i). Simultaneous DSA Grouping and Mask Assignment required for a DSA-MP process are shown in Figure 1c, and are the objective of the algorithms in this paper. It is required to determine the DSA groups and choose the mask for each group. Details about the hybrid DSA-MP process are presented in Section II.



(a) Grouping in DSA, resulting in four groups: one doublet and three singletons (b) Mask Assignment in Triple Patterning.



(c) Simultaneous DSA Grouping and Mask Assignment in a Hybrid DSA-MP process, using Double Patterning.

Fig. 1: Grouping required for DSA process, Mask Assignment required for MP process and Simultaneous DSA Grouping and Mask Assignment required for a DSA-MP process.

Our contribution in this work is summarized as follows:

- An optimal Integer Linear Programming (ILP) formulation is presented to perform the mask assignment and DSA grouping simultaneously, for a hybrid DSA-MP process in which all masks apply self-assembly. Heuristic algorithms are also proposed and are benchmarked against the ILP solution.
- An optimal ILP formulation is proposed to solve the mask assignment and DSA-grouping for a hybrid process in which not all the masks apply self-assembly. Heuristics are also presented to solve the same problem.

A. Multiple Patterning (MP)

The semiconductor industry has managed to scale to the 22nm node and beyond using MP [7], where pitch multiplication is achieved by using multiple masks to print one layer. There are two types of Multiple Patterning: the first one is based on N repetitions of Litho-Etch $(LE)^N$ where N is the number of masks, and the second is Self Aligned Multiple Patterning [7]. A lot of work has been done for mask decomposition for both types of MP, for example: [8], [9], [10], [11]. In this work, we assume a hybrid DSA-MP process which uses $(LE)^N$ along with DSA.

B. Brief Introduction to DSA

Self-Assembly is the phenomenon that occurs when block co-polymers composed of immiscible blocks phase-separate

Y. Badr is with the Department of Electrical Engineering, University of California, Los Angeles, USA e-mail: ybadr@ucla.edu.

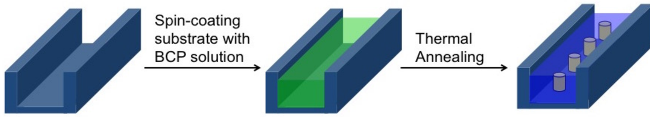


Fig. 2: An example DSA process of a diblock co-polymer using Graphoepitaxy

into organized structures [12]. For example, a diblock co-polymer can self-assemble into periodic structures of one type of block into a matrix of the other. Lithographically-printed patterns (in the Graphoepitaxy scheme) or chemically-treated surfaces (in the Chemoepitaxy scheme) are used to direct the self-assembly process and are called Guiding Templates. The graphoepitaxy process for contact holes is shown in Figure 2, where trenches are lithographically printed first, and then the surface is spin-coated with the block co-polymer (BCP). Upon thermal annealing, the phase separation occurs, and with a particular BCP and surface treatment of substrate [13], cylinders are obtained within the guiding template. Then one of the blocks is selectively etched, and the other block is used to transfer the pattern to the substrate underneath [14]. Thus pitch multiplication of the lithographically-printed patterns can be achieved. The realizable assembled pitch depends on the properties of BCP used.

C. DSA Capabilities

The BCP has a natural pitch L_0, to which it assembles, if not strongly guided by templates. To create a hole array with a pitch different from the natural pitch of the block copolymer, strong confinement is needed in the templates [15]. Small templates achieve strong lateral confinement for the block copolymer leading to more precise control of the self assembly process [4], [16]. In addition, smaller defects density can be obtained with smaller size of templates [17], [18], [19]. Accordingly templates should be designed such that a very small number of contacts are created per template [2]. In addition, previous research has reported that using peanut-shaped templates with a very narrow neck between every pair of contacts can lead to less placement error [2]. However well-modulated peanut shapes are hard to print in 193i photolithography, therefore it is preferred to have the pitch of grouped contacts close to the natural pitch of the copolymer and to avoid 2D groups altogether [2]. Diagonal groups (which have larger pitch than L_0) are also not desired because they need very strong confinement which can only be achieved by very complicated peanut-shape guiding templates which are also difficult to print in 193i photolithography [20].

D. CAD Flow for DSA

In a DSA process which prints the guiding templates in a single exposure, the guiding templates need to be determined based on the given contact/via layer. Figure 3 shows a typical flow that is used to design the guiding templates. First, the DSA grouping algorithm determines which contacts are to be assembled using the same guiding template and hence the DSA groups are generated. The grouping algorithm has to consider the lithography-driven spacing constraints which the guiding templates need to satisfy. For each DSA group, a guiding template is synthesized; the synthesis process attempts to reverse-engineer the self assembly process in order to come up with the correct templates. The templates then undergo the classical optical treatment like OPC and SRAF insertion to enhance the resolution. Finally verification is performed, to compare the expected assembled contacts (based on the synthesized templates) to the target contacts.

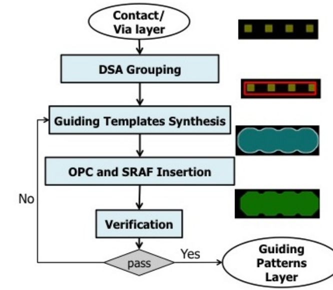


Fig. 3: CAD Flow for a Single-Exposure DSA Process

However in a technology that has multiple exposures, the DSA grouping method has to be coupled with the mask assignment method. In [21], it has been shown that cascading the traditional DSA grouping method with the Multiple Patterning Decomposer, which are both unaware of the hybrid nature of the process, produces poor results.

In this paper we study the optimal formulation and heuristic algorithms that can solve the DSA grouping and Mask assignment problem for two different schemes of the hybrid DSA-MP process.

The rest of the paper is organized as follows: section II describes the two schemes of the hybrid DSA-MP process assumed in this work and explains the rules of the hybrid process. In section III, we introduce the graph structure used in our algorithms. Sections IV and V present the optimal and the heuristic algorithms proposed for the two schemes of the hybrid DSA-MP process. Section VI shows and analyses the results. Finally, conclusions and future work are presented in section VII.

II. Hybrid DSA-MP Process

A. Alternative Hybrid Schemes

There are two alternative schemes for a hybrid DSA-MP process for contact/via holes[2]. In the first scheme, each of the N masks prints the guiding templates for DSA, then the self-assembly of the BCP will create the actual holes. We refer to this scheme as All_DSA. In the second scheme, some of the masks will directly print the contact holes and thus do not go through self-assembly, but the other masks will print the guiding templates and use self-assembly to create the holes. We refer to this scheme as $Partial_DSA$. The main advantage of the second scheme is that the masks bypassing DSA can print shapes or sizes different from the uniform dimensions determined by DSA, for example allowing rectangular (bar) vias which have appeared starting from the 28nm node [22].

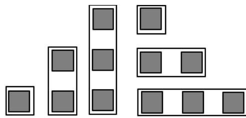


Fig. 4: Legal DSA groups, if maximum group size (max_g) is 3

In both schemes we assume that the same-mask minimum allowed pitch that applies to any two DSA groups has the same value as the same-mask minimum allowed pitch that applies between any two contacts/vias on a mask which is not applying self-assembly. This is the pitch we refer to as $litho_pitch$.

Under these assumptions, for a via/contact layer with **all** holes having the **same square dimension required by DSA**, the $Partial_DSA$ process scheme is more restrictive than the All_DSA scheme. This is proved in the next lemma.

Lemma 2.1: Any via layer which is compliant with $Partial_DSA$ scheme and whose vias are all squares of the same dimension determined by DSA is also manufacturable with the same number of masks in All_DSA.

Proof: Assume that there is a via layer that is manufacturable (violation-free) in $Partial_DSA$, but is not manufacturable with All_DSA, and all its vias are squares with the dimension compliant with DSA.

The conversion from $Partial_DSA$ to All_DSA is performed as follows: for each contact hole that has been assigned to a mask bypassing self-assembly, a guiding template is created on the corresponding mask in All_DSA, with one contact/via. All the masks that are applying self-assembly in $Partial_DSA$ are used in All_DSA without change.

Since there are no violations on the original masks for $Partial_DSA$, there must be no violations for the All_DSA masks resulting from the conversion above. Thus a contradiction exists, and this design must also be manufacturable in All_DSA. ■

However, designs which have bar vias in addition to the square vias may be manufacturable in $Partial_DSA$ but not in All_DSA. This is because self-assembly of the BCP can only result in holes with a particular uniform dimension. Holes of non-uniform dimensions can be patterned in $Partial_DSA$ by being assigned to the masks which do not apply self-assembly, assuming no coloring conflicts exist.

The work in this paper is only concerned with contact/via hole patterning. We assume that 193i is used to print the guiding templates. Accordingly, DSA-grouped contacts are only allowed to be collinear and either vertically or horizontally aligned, because the lithography variations in guiding templates needed for more complex DSA groups can lead to high defectivity level in self-assembly [23]. We assume there is a maximum number of allowed contacts per groups, which is an input value (max_g). For example, for a max_g value of 3, the legal DSA groups are the ones shown in Figure 4.

Thus, given a process which has Multiple Patterning (N masks) and DSA, it is required to do the DSA grouping and decompose the contact/via holes onto the N masks; in order to minimize the number of mask violations. In a violation-free solution, any two contacts/vias whose centers are separated by a distance less than the same-mask minimum pitch are either assigned to different masks or in the same DSA group on the same mask. Stitch-free decomposition has been assumed because most of the templates are expected to have small size.

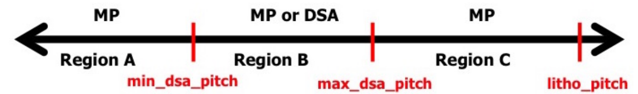


Fig. 5: Ranges of distance between two polygons, where DSA and/or MP can resolve the spacing conflict.

B. Important Parameters in the Process

There are several important parameters in this problem:

1) **Contact/Via hole DSA-manufacturable dimension** ($hole_dim$): the width of contact/via that is manufacturable though the self-assembly of the BCP. Only the square contacts/vias of this dimension are assumed to be DSA-manufacturable, while contact/via holes not adhering to this dimension can only be printed in the $Partial_DSA$ scheme.
2) **Minimum Grouping Pitch** (min_dsa_pitch): minimum distance that can exist between centers of two contact/via holes in a DSA group. This distance is equal to the natural pitch (L_0) of the block copolymer.
3) **Maximum grouping distance** (max_dsa_pitch): maximum distance that can exist between centers of two neighboring contacts/vias in one DSA group. This is derived from the properties of the block copolymer, because its self-assembly pitch can not be stretched beyond a certain threshold.
4) **Minimum Lithography Pitch** ($litho_pitch$): minimum space that can occur between the centers of any two shapes on a particular mask.
5) **Maximum DSA Group Size** (max_g): maximum number of holes that can be DSA-grouped together, and hence manufactured using the same guiding template.
6) **Number of masks** (N): number of masks/exposures in the process. We use b to denote the minimum number of bits required to encode N: $b = ceil[log_2(N)]$.

The distance range between centers of any two contacts/vias is divided into three regions, showing whether DSA grouping and/or assignment to different masks (MP) can be used to resolve the spacing violation, as shown in Figure 5 . Outside the DSA-allowed range [min_dsa_pitch, max_dsa_pitch], only MP can be used to resolve the conflict between the two contacts. Note that it has been assumed that $litho_pitch$ has larger value than max_dsa_pitch, which complies with the ranges in literature (for e.g. [4], [24]), assuming 193i lithography is used to print the templates.

III. Hybrid Grouping /Spacing Graph Representation

In this section, the new graph structure which considers both DSA and MP is explained. We use a hybrid grouping/spacing graph (GG/SG). Each contact/via is represented as a graph node. There are two types of edges: spacing edges, and grouping edges. A spacing edge exists between every two contacts whose centers are within $litho_pitch$ from each

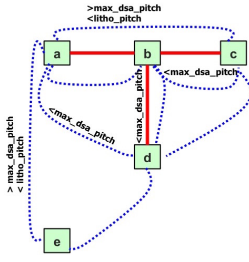


Fig. 6: Hybrid Graph between five contacts. Grouping edges are shown as solid lines, while spacing edges are shown as dotted curves. Distance between any two contacts/vias on this graph is at least greater than min_dsa_pitch. A spacing edge exists wherever the distance between the centers of two contacts/vias is less than $litho_pitch$ and a grouping edge exists wherever two direct-neighboring contacts can be grouped, i.e. distance between their centers is in the acceptable DSA grouping range, and are aligned on same X-axis or Y-axis.

other. A grouping edge is created between every two direct-neighboring contacts/vias that can be grouped into the same guiding template. These contacts/vias are aligned on the same horizontal or vertical axis, and the distance between their centers is within the DSA grouping interval: [min_dsa_pitch, max_dsa_pitch]. An example of the hybrid graph is shown in Figure 6. When we are only interested in the grouping edges, we refer to the graph as Grouping Graph (**GG**), and we refer to it as Spacing Graph (**SG**) [1] when only the spacing edges are of interest. Sections IV andV will show how the hybrid graph is used in the proposed algorithms.

Graph Division: Similar to [9], we apply the graph division technique based on connected components. The independent connected components of the hybrid graph are determined, and each independent component can be processed independently in the following algorithms. In our implementation, we use OpenMP threads in order to process the graph components in parallel.

IV. Algorithms for the All_DSA Scheme

In this section, we present the optimal ILP formulation and heuristic algorithms for the simultaneous DSA grouping and MP decomposition problem for the All_DSA scheme.

A. ILP Formulation

In a hybrid DSA-MP process, a **conflict** between two contacts/vias means that the distance between their centers is less than the $litho_pitch$, but are assigned to the same mask and they do not lie in the same DSA group. In this formulation, the objective is to minimize the number of conflicts. The constraints are derived from DSA as well as lithography requirements.

The constraints are generated based on the distance between centers of contacts/vias, according to the distance number line shown in Figure 5 which is summarized as follows: if the distance is less than min_dsa_pitch or greater than max_dsa_pitch, then the two contacts/vias have to to be assigned to different masks. If the distance is greater than min_dsa_pitch but less than max_dsa_pitch then the pair

[1]Note that the “Spacing Graph” is similar to the “Conflict Graph” used in some mask decomposition works, for e.g. [8].

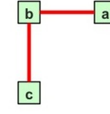


Fig. 7: Example for a connected component on GG. Contacts a and c do not have direct grouping edge, but a grouping path exists between them. Yet, they must not be grouped.

TABLE I: Notation used in ILP Formulation

m_i^k	k^{th} bit of mask index of ith contact/via
g_{ij}	Flag indicating if i^{th} and j^{th} contacts/vias are grouped
s_{ij}^k	Similarity variable indicating if k^{th} bits in masks of ith and j^{th} contacts/vias are identical
c_{ij}	Flag indicating if i^{th} and j^{th} contacts/vias are in conflict
SEs	set of spacing edges in GG/SG
GEs	set of grouping edges in GG/SG
$conn(i,j)$	Flag indicating if i^{th} and j^{th} contacts belong to same connected component in GG
$fg(i,j)$	Flag indicating if grouping of ith and j^{th} contacts is forbidden because they are not aligned or their inter-distance is not DSA-compliant (region A or C in Figure 5)
$ord(i,j,k)$	Flag indicating if i^{th} , j^{th} and k^{th} contacts are collinear and ordered, i.e. j^{th} contact lies between i^{th} and k^{th} contacts
$overlap(i,j,m,n)$	Flag indicating if group containing i^{th} and j^{th} contacts overlaps with group containing m^{th} and n^{th} contacts

of contacts/vias are either to be assigned to different masks or grouped together for DSA on the same mask. Otherwise, a conflict occurs.

To represent the problem in **linear** constraints, binary variables are used to encode the mask number, like [9]. Our ILP works for Double Patterning (DP), Triple Patterning (TP), Quadruple Patterning (QP) and other higher powers of two. However for simplicity of the notation, we only present it for QP, and we explain later the differences in the generated ILP when a different number of masks is used. For QP, two bits are required to represent the mask.

To generate the ILP constraints, it is required to construct the hybrid graph and then find the *connected components* in the GG. If two contacts/vias belong to the same connected component, then a path of grouping edges exists between them and therefore they can get grouped through that grouping path. However some of them may not be groupable because they are not manhattanly aligned, and thus their grouping has to be explicitly prohibited via special constraints. For example, in Figure 7, contacts a and b are allowed to be in same group, and contacts b and c can also be in same group, but these two simultaneous groupings imply the grouping of a and c which is disallowed because they are not manhattanly aligned. Therefore constraints must be added to prohibit grouping of non-groupable pairs that lie in the same connected component like the case of contacts a and c in Figure 7.

The variables and notation used are explained in Table I. The mathematical formulation is as follows:

$$minimize \quad \sum_i \sum_j c_{ij} \tag{1}$$

$$subject\ to:$$

$$s^1_{ij} + s^2_{ij} - g_{ij} \leq c_{ij} + 1 \quad \forall (i,j) \in \mathbf{SEs} \tag{2}$$

$$s^1_{ij} \geq 1 - m^1_i - m^1_j \quad \forall (i,j) \in \mathbf{SEs} \tag{3a}$$
$$s^1_{ij} \leq 1 - m^1_i + m^1_j \quad \forall (i,j) \in \mathbf{SEs} \tag{3b}$$
$$s^1_{ij} \leq 1 + m^1_i - m^1_j \quad \forall (i,j) \in \mathbf{SEs} \tag{3c}$$
$$s^1_{ij} \geq -1 + m^1_i + m^1_j \quad \forall (i,j) \in \mathbf{SEs} \tag{3d}$$
$$s^2_{ij} \geq 1 - m^2_i - m^2_j \quad \forall (i,j) \in \mathbf{SEs} \tag{3e}$$
$$s^2_{ij} \leq 1 - m^2_i + m^2_j \quad \forall (i,j) \in \mathbf{SEs} \tag{3f}$$
$$s^2_{ij} \leq 1 + m^2_i - m^2_j \quad \forall (i,j) \in \mathbf{SEs} \tag{3g}$$
$$s^2_{ij} \geq -1 + m^2_i + m^2_j \quad \forall (i,j) \in \mathbf{SEs} \tag{3h}$$

$$s^1_{ij} \geq g_{ij} \quad \forall (i,j) \in \mathbf{GEs} \tag{4a}$$
$$s^2_{ij} \geq g_{ij} \quad \forall (i,j) \in \mathbf{GEs} \tag{4b}$$

$$g_{ij} = 0 \quad \forall fg(i,j) = 1 \tag{5}$$

$$g_{ia} + g_{j_a} \leq 1 + g_{ij} \quad \forall conn(i,j) = 1, conn(i,a) = 1, conn(j,a) = 1 \tag{6}$$

$$\sum_{j, conn(i,j)=1, i \neq j} g_{ij} \leq max_g - 1 \quad \forall i \tag{7}$$

$$g_{ij} \leq g_{ia}, g_{ij} \leq g_{ja} \quad \forall ord(i,a,j) = 1 \tag{8}$$

$$g_{ij} + g_{mn} \leq 1 \quad \forall \{i,j,m,n | overlap(i,j,m,n) = 1\} \tag{9}$$

The objective function in Equation (1) aims at minimizing the number of conflicts.

Each of the constraints in Equation (2) is used to set the conflict variable between two contacts/vias having a spacing graph edge, if they are assigned to the same mask and are not DSA-grouped. Constraints in Equations (3a-3h) are linear representation of the XNOR boolean relationship between two mask bits (e.g. s^1_{ij}=m^1_i XNOR m^1_j) to set the similarity variable if the corresponding mask bits are identical. For example, for a pair of contacts i and j, if they are both assigned to mask 3, then $m^1_i = m^1_j = 1$ and $m^2_i = m^2_j = 1$ and if they are not grouped, then $g_{ij} = 0$. Accordingly, the similarity variables s^1_{ij} and s^2_{ij} are set to 1 due to Equations (3a-3h). As a result, Equation (2) sets the related conflict variable c_{ij} to 1, adding 1 to the cost function in Equation (1).

The constraints in Equations (4a) and (4b) ensure that any grouping variable between two contacts/vias can only be asserted if the two contacts/vias are assigned to the same mask. Constraints in Equation (5) disallow grouping of pairs of contacts/vias that do not satisfy DSA constraints. In addition constraints in Equation (6) impose the semantics of transitive grouping, i.e. if contacts x and a are grouped and contacts y and a are also grouped, then contacts x and y are grouped as a result. The constraints in Equation (7) enforce the maximum group size, since smaller group sizes have been found to lead to more robust assembly [16], [4], [2] due to better lateral confinement.

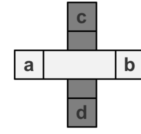


Fig. 8: Example of overlapping groups, (a, b) and (c, d); $overlap(a, b, c, d)$=1. If the two groups are assigned to two different masks, the selection of both groups can be forbidden or allowed, according to the requirements of the process.

The constraints in Equation (8) make sure that DSA groups are continuous, and not interleaving, meaning that if two contact/via holes can only be grouped if the contact/via lying between them is also in the same group. Finally the constraints in Equation (9) ensure that the groups which overlap in space are mutually exclusive, even if assigned to different masks. Figure 8 shows an example of overlapping groups (a, b) and (c, d), thus $overlap(a, b, c, d) = 1$. If overlap of groups is disallowed by the process, then the selection of these two groups simultaneously will not happen even if both groups are assigned to different masks, due to Equation (9). Note that even if overlap between groups on different masks is allowed by the process, these two groups will never be assigned to the same mask because the distance between the two groups is less than $litho_pitch$. The two sets of constraints of Equations (8) and (9) are optional; they depend on whether the process allows the overlap between templates on different masks. [2]

In case of Triple Patterning, one more constraint is required per polygon to prohibit using the unused mask combination [9], as shown in Equation (10).

$$m^1_i + m^2_i \leq 1 \quad \forall i \tag{10}$$

Note that for ease of understanding, the presented formulation hides some details which have been implemented to save memory. For example, the grouping variables are only created for pairs of contacts/vias which belong to the same connected component.

B. *Proposed Heuristics*

Since the optimal ILP does not scale to dense full-chip designs, we present heuristics to solve the decomposition and grouping problem efficiently. The objective is to try to resolve as many conflicts as possible by grouping or assignment to different masks. To achieve this objective, we use heuristics to maximize the chance of grouping in the whole contact layer, which is in return expected to maximize the possibility of being able to fix conflicts by DSA grouping. In other words, our objective is to find the biggest number of non-contradicting groupable pairs of contacts. These will be the candidate grouping options; a subset of which will be chosen by coloring. We propose the following two heuristics:

[2] In all our experiments, the overlap between templates on different masks is forbidden.

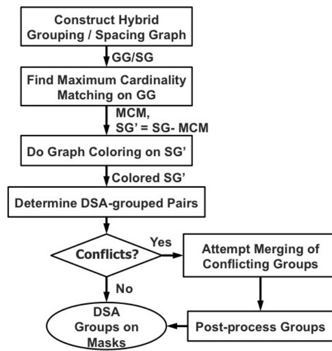


Fig. 9: The proposed DSA Grouping - MP Decomposition flow

*1) **Maximum Cardinality Matching Heuristic (MCM_H)**:* On the grouping graph, the problem of finding the biggest number of non-contradicting groupable pairs of contacts translates to finding the maximum number of grouping edges with no common nodes (i.e. contacts), since every grouping edge represents a grouping opportunity for the involved pair of contacts. This formulation is exactly the **Maximal Cardinality Matching** (MCM) problem, which finds the maximum number of disjoint edges and which can be solved in polynomial time using Edmond's algorithm [25]. We used an $O(m\,n\,alpha(m,n))$ implementation of the algorithm, where m is the number of edges, n is the number of vertices and $alpha(m,n)$ is the inverse Ackerman function and is upper bounded by 4.

The flow of the proposed algorithm is shown in Figure 9. At the beginning, the hybrid grouping/spacing graph structure described in section III is constructed. Then the maximum cardinality matching of the grouping graph is found. If the technology does not allow overlap between the templates on different masks, the MCM result is processed as follows: if the MCM result includes two pairs of matched vertices such that their resulting groups will overlap, one of the two pairs is arbitrarily chosen and removed from the MCM result. Then, the spacing edges between the matched vertices (i.e. the spacing edges that are identical to the grouping edges in the MCM result) are removed from the spacing graph [3], and we have a modified spacing graph SG'. The idea is to drop the spacing edges between polygons which can be grouped. When the Multiple Patterning decomposer is then run on SG', it need not assign these groupable contacts to different masks because they can be printed as one DSA group. Then, the matched pairs of contacts are processed; if they got assigned to the same mask, then a DSA group is created for them on their mask. If they got assigned to different masks, then each is left as a DSA group of a single contact on the mask it got assigned to.

Group Merging and Post-processing: The algorithm up to this point can only produce groups of singletons (only one contact/via in a group) and doublets at the largest. We attempt to resolve the remaining coloring conflicts by merging the conflicting groups if they satisfy the DSA constraints, thus tentatively creating groups larger than two. The merging is attempted as follows: a GG (defined in Section III) is constructed for the contacts/vias assigned to each mask separately. Connected components in this GG are then determined. In each connected component, if all the contacts/vias are collinear and the distance between the centers of all the neighboring vias are within the allowed DSA pitch range $[min_dsa_pitch, max_dsa_pitch]$), then one group is created for all these vias; otherwise the groups determined previously are unchanged and the merging does not happen for this connected component.

[3]Every grouping edge is also a spacing edge, because we assume max_dsa_pitch is smaller than $litho_pitch$.

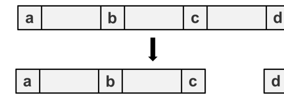


Fig. 10: Example of the group splitting step in MCM_H, assuming max_g=3. A group of four gets split into two neighboring groups of size three and one. .

The merging step can result in groups exceeding the maximum allowed number of contacts/vias per group (max_g). Thus, a post-processing step is needed in which each such "large" group whose size exceeds max_g is split into two or more groups as follows: the contacts/vias in a "large" group are sorted according to their left edge coordinate (if it is a horizontal group), or bottom edge coordinate (if it is a vertical group). New groups are created for the sorted contacts resulting in groups of size max_g and possibly one group of size smaller than max_g. Note that a conflict will exist between every two neighboring groups resulting from the split. An example of group splitting is shown in Figure 10, where a group of four gets split into a triplet and a singleton because $max_g = 3$.

***Example**:* Figure 11 shows an example of the flow on a layout snippet. First, the hybrid grouping/spacing graph (GG/SG) is constructed where the red edges are the grouping edges and the blue edges are the spacing edges. Then, the MCM solution (not unique in this example) is computed, where it is found to consist of the six edges: a - b, b - c, c - d, e - f, c - e and d - f. After removing the spacing edges corresponding to the MCM solution, the modified spacing graph SG' is obtained. The graph is then colored using two colors (for Double Patterning), in this case assigning vias a, e and f to one of the masks and assigning vias b, c and d to the other mask. Since the matched contacts c and d were assigned to the same mask, they form a DSA group together. Also vias e and f form a DSA group. One conflict remains between vias b and c, and thus the singleton group of b is merged with the group of c and d. The final grouping and decomposition result is: a singleton group containing via a, a group of e and f on a mask; and a group of b, c and d on the other mask. None of the resulting groups contains more vias than the maximum allowed (max_g), which was assumed to be four in this example, so the post-processing step is not needed in this example and is not shown in Figure 11.

*2) **A Trivial Heuristic** (Trivial_H):* In some cases, a much simpler heuristic algorithm can be used. This is to drop all spacing edges that coincide with grouping edges. The rest of the flow of Figure 9 remains the same except that MCM is not computed, and the modified spacing graph SG' is created as $SG - GG$; i.e. all the spacing edges coinciding with grouping

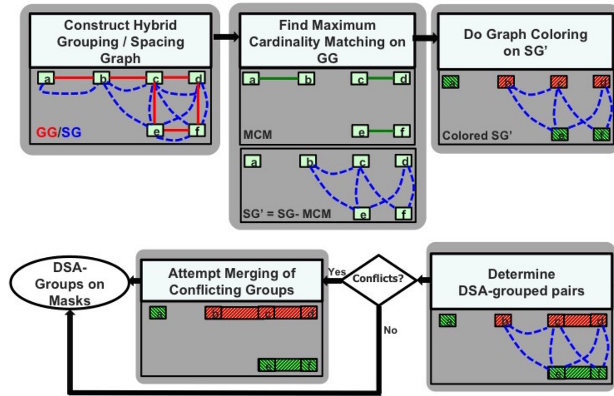


Fig. 11: Example of the proposed heuristic algorithm MCM_H. The post-processing step is not needed in this example and is not shown. .

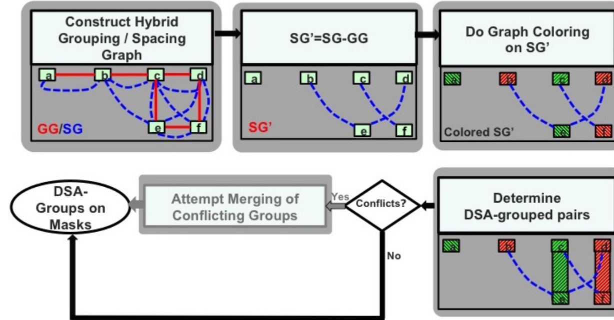


Fig. 12: Example of the trivial heuristic algorithm Trivial_H. The post-processing step is not needed in this particular example and is not shown. .

edges are removed.

Figure 12 shows an example of the flow for the trivial heuristic on a layout snippet. As shown in the figure, all spacing edges that coincide with grouping edges are removed from the spacing graph, in order to create the modified spacing graph. The rest of the flow is the same as the one shown in the example of Figure 11.

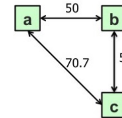
If the technology disallows overlap between the templates on different masks (Figure 8), one of every two grouping edges that correspond to overlapping groups is arbitrarily chosen and deleted (the grouping edge is deleted but the corresponding spacing edge is not deleted).

However, this heuristic requires a constraint to make sure that by dropping these spacing edges, there is no chance of ending up with illegal groups. [4] Let $litho_dist$ be the minimum allowed distance between two vias on the same mask, which is equivalent to $litho_pitch - hole_dim$, and let max_dsa_dist be the maximum allowed distance between two vias in the same DSA group, which is equivalent to $max_dsa_pitch - hole_dim$. The required constraint is then:

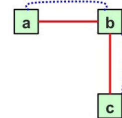
$$litho_dist > \sqrt{2}\; max_dsa_dist \tag{11}$$

If the rule values violate this constraint (for example,

[4] Remember that the allowed groups in this work are collinear manhattan groups only.



(a) Layout snippet with which the trivial heuristic may fail



(b) Corresponding hybrid graph

Fig. 13: Example of a layout snippet with which the trivial heuristic may fail, and its corresponding hybrid graph, assuming max_dsa_pitch=64 and $litho_pitch$=84

max_dsa_dist=50 and $litho_dist$=70), this trivial heuristic can not be used because it can lead to illegal groups. For example, this trivial heuristic may fail if run on the snippet shown in Figure 13, because all the spacing edges will be dropped and hence one possible solution is to assign the three vias to the same DSA group (and the same mask), and result in an L-shaped DSA group, which is illegal. However, if the constraint in Equation 11 had been satisfied by the values of max_dsa_dist and $litho_dist$, there would have been a spacing edge between shapes a and c, preventing their being assigned to the same mask, and accordingly they would not be grouped.

In comparison to MCM_H, this heuristic performs the graph coloring on a simpler, less constrained graph, since all the spacing edges coinciding with grouping edges are removed, while MCM_H only removes the spacing edges that

coincide with the MCM result (MCM result is subset of GG). Thus, Trivial_H can result in fewer conflicts. However, one disadvantage is that in Trivial_H, the graph coloring algorithm can unnecessarily result in big groups, potentially exceeding max_g vias, since there is no notion of maximum group size constraint, once the modified SG is used as input to the graph coloring algorithm. Each group with number of contacts/vias exceeding max_g gets split into smaller groups on the same mask, having spacing violations among them, in the post-processing step. This can lead to unnecessary conflicts in the result of the Trivial_H. However, this is not a problem in MCM_H, because the initial result (before the conflicting group-merging attempt) of MCM_H contains groups of two vias at most, thus if it results in a zero-conflict solution before group-merging , it is guaranteed to produce a final conflict-free solution.

V. Algorithms for the $Partial_DSA$ Scheme

The $Partial_DSA$ process described in Section II can allow more flexibility in the dimensions of the manufacturable contacts/vias. In such a process, some masks will be used to print the via/contact holes directly and will skip the self-assembly step. These masks can be used to print the holes whose dimensions are not compliant to DSA. We refer to these masks as *special masks* and to the vias/contacts not having the required DSA dimensions as *special holes*, i.e. contact/vias which are not squares or whose dimension is not equal to $hole_dim$. The holes having the required DSA dimensions are referred to as *regular holes*. Note that the methods described in Section IV can not be used as is, because the regular vias can also be patterned on the *special masks* without groups; otherwise, the special masks are not fully utilized and unnecessary violations can exist on the *non-special masks*. This is why, the problem can not be solved by assigning all special holes to *special masks* and using the same *ALL_DSA* methods for the regular holes on the *non-special masks* only.

In this section, we present the optimal ILP formulation and heuristic algorithms for the simultaneous DSA grouping and MP decomposition problem for the $Partial_DSA$ scheme.

A. ILP Formulation

The same variables and notations shown in Table I are used. In addition, we use **SH** to refer to the set of *special holes*.

The cost function and constraints explained in Section IV-A are used in this ILP formulation. However a few constraints are added in order to model the special case of the $Partial_DSA$ process. Namely, special mask assignment constraints and special grouping constraints are added. The special mask assignment constraints ensure that each *special hole* can only be assigned to one of the *special masks*. The special grouping constraints prohibit the grouping of holes assigned to the *special masks* since these masks will not apply self-assembly. We assume that a process can have one or two *special masks*, which will be mask 0 or masks 0 and 1, respectively. We

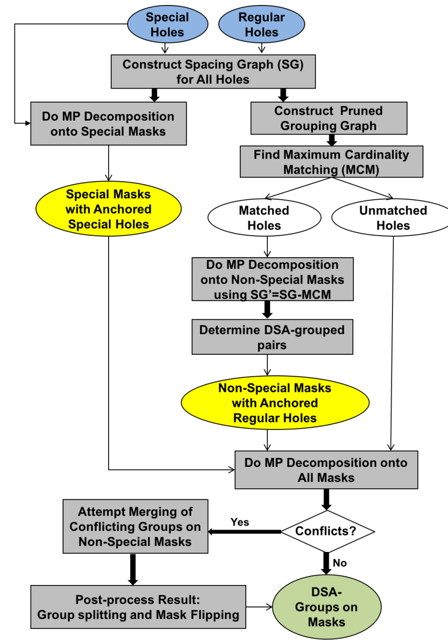


Fig. 14: Flow of the Partial_MCM_H: MCM-based heuristic for the $Partial_DSA$ scheme

show the constraints here assuming four masks are used, for simplicity of the notation.

$$m_i^1 = 0 \quad \forall\, i \in \mathbf{SH} \tag{12a}$$
$$m_i^2 = 0 \quad \forall\, i \in \mathbf{SH} \tag{12b}$$
$$m_i^2 = 0 \quad \forall\, i \in \mathbf{SH} \tag{13}$$
$$g_{ij} \leq m_i^1 + m_i^2 \; \forall\, (i,j) \in GEs \tag{14}$$
$$g_{ij} \leq m_j^2 \quad \forall\, (i,j) \in GEs \tag{15}$$

The mathematical formulation of the special mask assignment constraints in case of one *special mask* only are shown in Equation (12). In this case, all *special holes* must be assigned to mask 0. In case of two *special masks*, each *special hole* can either be assigned to mask 0 (00_2) or mask 1 (01_2), thus the least significant bit (m_i^1) is not constrained. The constraint in this case is shown in Equation (13).

In case of one *special mask* only, the special grouping constraints are added in order to allow grouping only for holes assigned to non-special masks (masks 2 (10_2) and 3 (11_2)), as shown in Equation (14). If a hole is assigned to mask 0 (00_2), which is a *special mask*, then all its grouping variables are forced to 0. In case of two *special masks*, the special grouping constraints are shown in Equation (15), to allow grouping to happen only if the involved holes are assigned to mask 2 (10_2) or mask 3 (11_2)), which are the non-special masks .

B. Proposed Heuristics

We propose two heuristic flows which can perform the DSA grouping and mask assignment for a via/contact holes layer, knowing that no self-assembly will take place on the *special masks*.

*1) **MCM Heuristic for Partial_DSA (Partial_MCM_H)**:* The first proposed flow is shown in Figure 14. First, we construct a spacing graph to represent the contact/via layer. The spacing graph for *Special holes* only is colored using the special masks only.

Next, we create a *pruned* grouping graph of the regular holes; a pruned grouping graph is a grouping graph which excludes some of the vias for which grouping is not essential to remove conflicts. Avoiding unnecessary grouping is a useful heuristic in $Partial_DSA$ because grouped holes can only be assigned to one of the *non-special masks*; while non-grouped holes can be assigned to any of the masks, resulting in higher flexibility and accordingly less conflicts. We use the fact that if a graph is N-colorable, then adding a node with degree (number of edges) less than N makes the new graph N-colorable too [26]. Thus, similar to the layout graph simplification in [9]; starting from the spacing graph of the complete layer, we perform recursive deletion of nodes representing *regular holes* with degree less than the number of masks N, but we do not delete nodes which are direct neighbors of *special holes*). This is because *special holes* have an additional constraint that they can only be assigned to *special masks*, and accordingly the coloring options for the neighbors of *special holes* can be limited, and grouping can be the only way to resolve their conflicts in some cases.

Grouping edges are added for the remaining holes only. The resulting graph is the *pruned* grouping graph. MCM is then computed on it. The edges in the MCM solution are then removed from the spacing graph of the matched holes (holes which have incident edges in the MCM). The resulting modified spacing graph (SG') of the matched holes is then provided as input to a Graph Coloring algorithm, in order to assign the matched holes to the *non-special masks* only. This is because the matched contacts are the candidates for being DSA-grouped, and special masks will not have self-assembly. After the coloring, a DSA group is created for each pair of matched holes that have been assigned to the same mask.

At this point, each *special hole* has been anchored to a *special mask* and each regular hole involved in the MCM result has been anchored to one of the *non-special masks* and potentially grouped. The remaining non-assigned holes are then decomposed onto **all** the masks, which will have some anchored holes from the earlier steps. Then we attempt to resolve any remaining conflicts on the *special masks* by merging adjacent DSA groups, in the same way that has been explained in Section IV-B1.

Post-processing: Group Splitting and Mask Flipping: Finally the result is post-processed with two objectives. First, groups with more than max_g holes are split into two or more groups, in the same method explained in Section IV-B1. Second, we attempt mask flipping to resolve conflicts. This is especially important because before this step, the matched holes were only allowed to be assigned to the *non-special masks* in order to be DSA-grouped if needed, but these holes can alternatively be assigned to the *special masks* without grouping. Thus, the mask of some contact/via holes can be changed in the post-processing step to reduce the number of conflicts.

Algorithm 1 Group Splitting and Mask Flipping for $Partial_DSA$

Require: $break_groups$ (a boolean flag to allow breaking down groups to decrease conflicts)
1: **for** $m = 1$ to number of masks **do**
2: **for** group g in groups[m] **do**
3: **if** $size(g) > max_g$ **then**
4: split g into groups of size max_g or smaller (Section IV-B1)
5: **else**
6: **if** $size(g) = 1$ **then**
7: Attempt Mask Flipping ($g[1]$)
8: **else if** $break_groups$ or size of $g = 1$ **then**
9: Attempt Mask Flipping ($g[1]$)
10: Attempt Mask Flipping ($g[size(g)]$)

Algorithm 2 Attempt Mask Flipping

Require: $contact$
1: $min_conf \leftarrow$ number of conflicts of $contact$ on $mask[contact]$
2: **for** $m = 1$ to number of masks **do**
3: $n_conf \leftarrow$ number of conflicts of $contact$ on mask m
4: **if** $n_conf <$ number of conflicts of $contact$ on $mask[contact]$ **then**
5: $mask[contact] \leftarrow m$
6: $min_conf \leftarrow n_conf$
7: **if** $min_conf > 0$ **then**
8: **for** neighbor n in $neighbors[contact]$ in SG **do**
9: **if** n is not grouped **then**
10: $curr_conf \leftarrow$ number of conflicts of n on $mask[contact]$
11: **if** $curr_conf = 0$ **then**
12: n_conf =number of conflicts of $contact$ on $mask[n]$
13: **if** $(n_conf - 1) < min_conf$ **then**
14: swap($mask[contact]$, $mask[n]$)
15: $min_conf \leftarrow n_conf$

The algorithms used in the group splitting and mask flipping algorithm are shown in Algorithms 1 and 2. Each group containing more than max_g holes is split into smaller groups as explained in Section IV-B1. If a group is a singleton, then the mask flipping algorithm (Algorithm 2) is executed on it. If a group is not a singleton but has fewer holes than max_g and it is desired to break down groups for the sake of decreasing conflicts ($break_groups = 1$ in Algorithm 2), then the group gets broken and the two holes at both ends of the group undergo mask flipping. We choose not to attempt mask flipping on holes lying between two other holes in the same group, since removing such a hole can render the group invalid because the distance between the centers of the two other holes can exceed the max_dsa_pitch, and it may also lead to overlap between groups on different masks which is prohibited in some processes.

Given, a particular contact, the mask flipping algorithm (Algorithm 2) first tries to re-assign the contact to the mask where it would have the fewest number of conflicts, without changing the mask assignment of the other contacts. Finding the number of conflicts of the contact on a mask includes the effect of grouping the contact with the neighboring groups on the new mask. If the number of conflicts of the contact is not zero, then we attempt a mask exchange between the contact and its neighbors in SG). Swapping masks does not take place unless it will result in zero conflicts for the neigboring contact, and the neighboring contact was not grouped with other contacts. Several iterations of the whole flow of Algorithm 1 can be executed, adding to runtime. In practice we found that three iterations are usually enough, first with the $break_groups$ flag set to false in order to attempt resolving the violations with minimum change possible first. In the second iteration, we set $break_groups$ to true and in the third, we set it to false.

*2) **A Trivial Heuristic for Partial_DSA (Partial_Trivial_H)**:* The trivial heuristic proposed in Section IV-B2 can also be

TABLE II: Number of vias in test cases used in Experiments for All_DSA

Test case	Number of Vias
AES	48123
CortexM0	35255
LEON3	93474
MIPS	34784

TABLE III: Parameter Values (in nm) used in Experiments

min_dsa_pitch	34
max_dsa_pitch	56
$litho_pitch$	80
max_g	4
$hole_dim$	14
L_0	34
N	2 (DP) and 3 (TP)

applied to the $Partial_DSA$ scheme, by following the same flow shown in Figure 14 with few changes. Instead of finding the MCM on the *pruned* grouping graph and removing all the matched edges from the spacing graph, all the grouping edges in the *pruned* grouping graph are deleted from the spacing graph to obtain the modified spacing graph. All shapes having grouping edges are considered as matched holes, and the rest are considered as unmatched holes. The rest of the flow is the same. The same constraint explained in Section IV-B2 must be satisfied by the rule values, in order to be able to apply this heuristic.

VI. EXPERIMENTS AND RESULTS

A. Experimental Setup

The ILP and the heuristics were implemented in C++. Open Access was used for layout manipulation; IBM CPLEX was used to solve the ILP and Boost Graph API was used for graph operations. In addition, Mentor Graphics Calibre tool was used for Multiple Patterning decomposition (Graph Coloring). All the experiments were run on a shared cluster [27], using four cores and using up to 70GB of memory.

The test cases we use are: AES and MIPS from [28], ARM Cortex M0 processor and LEON3 Sparc processor. These have been synthesized, placed and routed, as will be mentioned later. The experiments were performed on the Via1 layer of the layouts.

B. Results for the All_DSA Scheme

The used layouts for the experiments for the All_DSA scheme have been synthesized, placed and routed using commercial 45nm SOI libraries, then sized and scaled. After modification of the layouts, the via width is 14nm and the minimum spacing is 21nm, which is close to ITRS contact pitch for 2025. The number of vias in each test case is shown in Table II.

The values that we use for our experiments are shown in Table III, unless noted otherwise for particular experiments.

1) ***Comparison between Different Approaches:*** We compare the following approaches:

ILP: the ILP formulation explained in Section IV-A
MCM_H: the MCM heuristic explained in Section IV-B1
Trivial_H: the trivial heuristic explained in Section IV-B2
MP_GP: MP decomposition followed by DSA grouping on each separate mask[21]
GP_MP: DSA grouping followed by MP decomposition[21]
DP only: DP decomposition, without DSA, using Calibre Double Patterning tool
TP only: TP decomposition, without DSA, using Calibre Triple Patterning tool

MP_GP and GP_MP are the two simple sequential approaches discussed in [21]. These two approaches have been implemented by using the conventional Calibre Multiple Patterning and Directed Self Assembly tools, which are not aware of the process being hybrid DSA-MP.

In Tables IV and V, we show the number of spacing violations between the groups in each of the layouts for different approaches. The runtime shown for MCM_H includes the complete flow in Figure 9. MCM_H has a 17% increase in the total number of violations for DP and TP and has a speedup of 4x , in comparison to the ILP solution. The average group size for MCM_H is 1.026 vias and 1.014 vias for DP and TP respectively, which means that most of the vias ended up in singletons and thus are expected to result in relatively high yield [2]. The average group size in the ILP solution is 1.016 vias and 1.007 vias for DP and TP respectively, even though the ILP cost function does not minimize the group size. In addition, MCM_H outperforms produces 56% fewer violations (in total) than GP_MP and MP_GP.

Trivial_H has a 5% increase in violations in comparison to the ILP, and has speed of 5x. The average group size for this heuristic is 1.0273 vias for DP and 1.015 vias for TP.

The trivial heuristic did outperform MCM_H in these test cases. As discussed in Section IV-B2, this is because Trivial_H removes all the spacing edges that coincide with grouping edges before the graph coloring, whereas MCM_H only removes at most one spacing edge for every shape. Thus, the graph coloring algorithm needs to solve a less constrained problem (graph with fewer edges) in case of Trivial_H. Although Trivial_H is expected to form bigger groups, the average group size generated by both MCM_H and Trivial_H are found to be approximately equal. This is because with the used rule values, the spacing graph is non-planar, i.e. spacing edges existed between vias which are not direct neighbors, and since grouping edges are between direct neighbors only, these spacing edges were not removed. Accordingly, the coloring algorithm assigned these non-direct-neighboring vias to different masks, creating small groups as a result, instead of big groups on the same mask. However, as mentioned before the rules need to satisfy the constraint explained in Equation(11) in order to be able to use this trivial heuristic, and the set of rule values in this experiment satisfies this constraint. Note that in Tables IV, V and VIII only, we ran the heuristics MCM_H and Trivial_H using one thread, since the single-threaded execution for them was already fast enough, that the overhead of thread management did add to the runtime, but the ILP is run using four OpenMP threads.

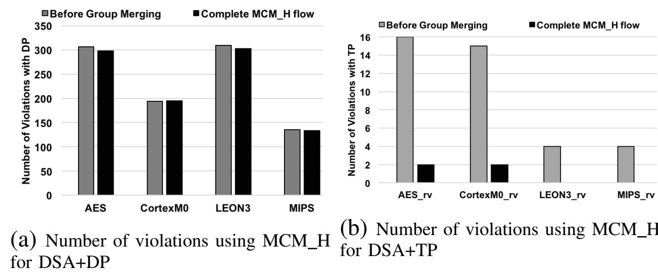
These results are expected to be pessimistic since the layouts are not optimized for DSA. Since technology and design are usually co-optimized, we should expect more DSA-friendly layouts.

TABLE IV: Results on Layouts with DP, for the All_DSA scheme

	DSA + DP: ILP		DSA+DP: MCM_H		DSA+DP: Trivial_H		DSA+DP: GP_MP		DSA+DP: MP_GP		DP only
	Violations	Time (s)	Viol.	Time (s)	Violations	Time (s)	Violations	Time (s)	Violations	Time (s)	Viol.
AES	257	59	298	25	291	21	696	1	641	1	815
CortexM0	154	33	195	12	128	21	487	1	488	1	671
LEON3	268	219	303	12	280	11	680	1	642	1	779
MIPS	115	59	133	49	131	32	324	1	315	1	391
Comparison	0.96	4.35	1.12	1.15	1.00	1.00	2.63	0.05	2.51	0.05	3.20

TABLE V: Results on Layouts with TP, for the All_DSA scheme

	DSA + TP: ILP		DSA+TP: MCM_H		DSA+TP: Trivial_H		DSA+TP: GP_MP		DSA+TP: MP_GP		TP only
	Violations	Time (s)	Violations	Time (s)	Violations	Time (s)	Violations	Time (s)	Violations	Time (s)	Viol.
AES	2	87	2	24	2	25	29	1	6	1	29
CortexM0	1	27	2	8	1	19	28	1	7	1	28
LEON3	0	207	0	24	0	17	7	1	1	1	7
MIPS	0	0 62	0	30	1	31	13	1	5	1 1	3
Comparison	0.75	4.16	1.00	0.93	1.00	1.00	19.25	0.04	4.75	0.04	19.25



(a) Number of violations using MCM_H for DSA+DP

(b) Number of violations using MCM_H for DSA+TP

Fig. 15: Number of violations using MCM_H before group merging as well as at the end of the complete MCM_H flow, for DSA+DP and DSA+TP

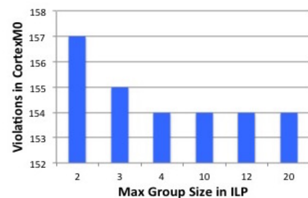


Fig. 16: Number of Violations vs. max_g in ILP in CortexM0 testcase with DP, for the All_DSA scheme

In Figure 15, we show the number of violations at two points in the flow of MCM_H: before group merging and splitting, as well as the end of the whole flow shown in Figure 9. The group merging and splitting steps does help reduce the number of violations significantly for DSA+TP case.

2) ***Change in the Number of Violations with the Maximum Group Size:*** In Figure 16, we show how the number of violations from ILP changes as the maximum group size changes, on CORTEXM0 in DP. The size of the formed groups did not exceed four. Moreover, by restricting the maximum group size to two and three contacts per template, a 1.9% and a 1.3% increase in the number of violations are acquired respectively, which is a small penalty, given that the assembly process is more robust for small groups.

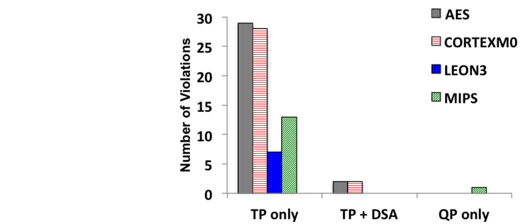


Fig. 17: Number of Violations with TP only, TP + DSA using MCM_H, QP only, for the All_DSA scheme

3) ***Possibility of Replacing a Mask, by using DSA :*** Another experiment was performed in order to assess the effectiveness of the assumed hybrid DSA-MP process. In Figure 17, the number of violations for each of the complete layouts is shown as a result of doing TP only, TP with DSA using MCM_H and finally QP only. Without the use of DSA, it is likely that QP is needed, leading to a higher cost process. However with DSA, at most two violations existed in each test case and these violations are likely to be eliminated when DSA-friendly design rules and layouts are available.

C. Results for the $Partial_DSA$ Scheme

In this section we show results for the second DSA-MP integration scheme which is $Partial_DSA$. To evaluate the algorithms for $Partial_DSA$ we synthesized, placed and routed the same designs with a 28nm library which has rectangular vias, in addition to square vias. The resulting layouts were sized and scaled. After modification of the layouts, the via width is 14nm and the minimum spacing is 20nm. Moreover, to be able to benchmark the heuristic against the optimal ILP, a snippet was used from each layout because otherwise the ILP either did run out of memory or did not finish within 12 hours using four threads.

The number of vias in each snippet and complete layout is shown in Table VI. The test cases are named in this way because "rv" indicates that these test cases have rectangular vias, "s" indicates that a snippet of the layout is used and "srv" indicates that the layouts have sparser usage of rectangular vias

TABLE VI: Number of vias in test cases used in Experiments for $Partial_DSA$

Test case	Num. of Vias	Num. of Rect. Vias	Num. of Square Vias
AES_rv_s	1551	239	1312
CortexM0_rv_s	1758	246	1512
LEON3_rv_s	1608	178	1430
MIPS_rv_s	1691	204	1487
AES_rv	124451	16585	107866
CortexM0_rv	107481	14295	93186
LEON3_rv	374993	47794	327199
MIPS_rv	129659	17749	111910
AES_srv_s	7835	11	7824
CortexM0_srv_s	2630	105	2525
LEON3_rv_s	2678	126	2552
MIPS_srv_s	8296	384	7912
AES_srv	116974	2457	114517
CortexM0_srv	98792	3053	95739
LEON3_srv	340156	21943	318213
MIPS_srv	119238	5953	113285

TABLE VII: Parameter Values (in nm) used in Experiments for $Partial_DSA$

min_dsa_pitch	33
max_dsa_pitch	56
$litho_pitch$	80
max_g	4
$hole_dim$	14
L_0	34
N	4 (QP)

than “rv”. The used rule values are shown in Table VII.

*1) **Comparison between Different Approaches**:* We use Partial_MCM_H to refer to the flow described in Section V-B1 and Partial_Trivial_H to refer to the trivial heuristic described in Section V-B2. Both are compared to the optimal ILP in Table VIII with respect to the number of violations and runtime. We assume a process with Quadruple Patterning where only two of the four masks apply self-assembly (i.e. two masks are *special masks*). The number of violations is counted between the DSA-groups on the *non-special masks* and between the via holes on the *special masks*. Partial_MCM_DSA has an 18% increase in the number of violations in comparison to the ILP solution and has a speedup of 23x, and an average group size of 1.33 vias on the *non-special masks* . Partial_Trivial_H also has 14% increase in the number of violations and has a speedup of 29x speedup and an average group size of 1.69 vias on the *non-special masks*. The average group size on the *non-special masks* for ILP is 1.21 vias on the *non-special masks*. In the Partial_DSA scheme, the trivial heuristic did outperform MCM_H, and did not have larger group size, due to the same reasons explained in Section VI-B1.

We also show results of the sequential approaches Partial_GP_MP and Partial_MP_GP in Table VIII. In Partial_GP_MP, the *regular holes* are grouped using Calibre DSA tools. The *special holes* are multi-patterned onto the *special masks* only and the grouped *regular holes* are multi-patterned onto the *non-special masks* and anchored. Finally the *regular holes* which were not grouped are multi-patterned onto all the masks. Whereas in Partial_MP_GP, the *special holes* are multi-patterned on the *special masks* only, and the *regular holes* are multi-patterned onto all the masks. The *regular holes* which get assigned to each of the *special masks* undergo DSA grouping. The number of violations using QP only without DSA (done using Calibre QP tool) is also shown in Table VIII.

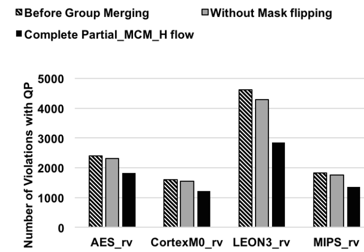


Fig. 18: Number of violations using Partial_MCM_H before group merging, after group merging and splitting and finally using the complete flow.

The two heuristics Partial_MCM_H and Partial_Trivial_H as well as the sequential approached Partial_GP_MP and Partial_MP_GP have been run on four complete test cases. The results are shown in Table IX, along with results of QP decomposition, without DSA, using Calibre QP tool. The average group size on the *non-special masks* for $Partial_MCM_H$ on the complete test cases is 1.39 vias, while that of $Partial_Trivial_H$ is 1.38 vias.

The big number of violations is attributed to two closely-related reasons. First, these layouts are not DSA-aware. Second, there is a lot of rectangular vias in the layouts, and there are cliques of four rectangular vias in the layouts requiring four masks, as opposed to two *special masks* only in these experiments. This is why QP without DSA is more appropriate for these layouts.

As shown in Table IX, Partial_Trivial_H sometimes has longer runtime than Partial_MCM_H in relatively dense test cases like LEON3 where the majority of the vias lie in the same connected component of the graph. This is because the process of checking for overlap between candidate groups takes more time as the number of candidate groups increases.The candidate groups in Partial_Trivial_H are all the grouping edges in the pruned grouping graph, while in Partial_MCM_H the candidate groups are the MCM result set only. [5]

Figure 18 shows the number of violations using Partial_MCM_H before group merging; after grouping merging and splitting but without mask flipping; and using the complete flow Partial_MCM_H. The group merging and mask flipping improve the quality of the heuristics by decreasing violations.

*2) **Results with Sparse Usage of Rectangular Vias**:* The density multiplication feature of DSA can not apply to rectangular vias. Thus in order to print rectangular vias using a DSA-based technology, larger spacing design rules need to be enforced between rectangular vias. To empirically test this claim, we synthesized layouts after increasing the minimum space design rule between rectangular vias by 87% and did not instruct the router to exert high effort in using the rectangular vias. The resulting layouts have fewer rectangular vias than the ones used in Tables IX and VIII, as shown in Table VI.

The results are shown for the clips in Table X and for the complete layouts in Table XI. With the sparse usage of rectangular vias, Partial_DSA can indeed help increase the via density in comparison to QP; QP does produce a

[5] The process of checking overlap can be made faster by using a spatial hashing technique similar to the method in [29].

TABLE VIII: Results on Layout Clips with QP, for the $Partial_DSA$ scheme. p_s is percentage of contacts assigned to special masks.

	Partial_DSA:ILP			Partial_MCM_H			Partial_Trivial_H			Partial_GP_MP		Partial_MP_GP		QP only
	Viol.	Time(s)	p_s (%)	Viol.	Time(s)	p_s (%)	Viol.	Time(s)	p_s (%)	Viol.	Time(s)	Viol.	Time(s)	Viol.
AES_rv_s	22	78	51	26	15	47	25	7	46	29	1	31	1	1
CortexM0_rv_s	30	54	51	37	11	50	35	8	52	42	1	44	1	2
LEON3_rv_s	8	378	47	8	9	40	8	10	39	19	1	9	1	0
MIPS_rv_s	5	416	46	6	5	39	6	7	50	18	1	21	1	0
Comparison	0.88	28.94	1.04	1.04	1.25	0.94	1.00	1.00	1.00	1.46	0.13	1.42	0.13	0.04

TABLE IX: Results on Complete Layouts with QP, for the $Partial_DSA$ scheme. p_s is percentage of contacts assigned to special masks.

	Partial_MCM_H			Partial_Trivial_H			Partial_GP_MP		Partial_MP_GP		QP only
	Viol.	Time(s)	p_s (%)	Viol.	Time(s)	p_s (%)	Viol.	Time(s)	Viol.	Time(s)	Viol.
AES_rv	1827	141	46	1835	136	45	2174	4	2286	3	60
CortexM0_rv	1214	268	46	1216	127	45	1543	4	1624	3	22
LEON3_rv	2842	881	38	2830	1962	38	5677	15	6264	7	93
MIPS_rv	1362	143	44	1350	141	43	1952	4	2067	3	15
Comparison	1.00	0.61	1.02	1.00	1.00	1.00	1.57	0.01	1.69	0.01	0.03

TABLE X: Results on Layout Clips with Sparse Rectangular Vias with QP, for the $Partial_DSA$ scheme. p_s is percentage of contacts assigned to special masks.

	ILP			Partial_MCM_H			Partial_Trivial_H			Partial_GP_MP		Partial_MP_GP		QP only
	Viol.	Time(s)	p_s (%)	Viol.	Time(s)	p_s (%)	Viol.	Time(s)	p_s (%)	Viol.	Time(s)	Viol.	Time(s)	Viol.
AES_srv_s	0	8922	47	2	14	49	0	16	49	5	2	6	2	5
CortexM0_srv_s	0	2335	43	1	10	43	0	13	43	3	2	3	2	3
LEON3_rv_s	0	2607	41	2	14	43	0	14	42	3	1	4	2	1
MIPS_srv_s	1	4882	47	3	15	44	2	17	43	28	2	29	2	0
Comparison	0.50	312.43	1.00	4.00	0.88	0.52	1.00	1.00	1.00	19.50	0.12	21.00	0.13	4.50

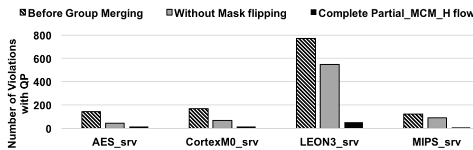


Fig. 19: Number of violations using Partial_MCM_H before group merging; after group merging and splitting but without mask flipping; and using the complete flow, on the complete layouts with sparse rectangular vias.

number of violations which is 8 times larger than that of Partial_Trivial_DSA on the complete layouts.

Figure 19 shows the number of violations using Partial_MCM_H before group merging; after grouping merging and splitting but without mask flipping; and using the complete flow of Partial_MCM_H, on the layouts having sparse rectangular vias.

VII. Conclusion

In this paper, we presented algorithms to solve the simultaneous DSA grouping an MP Decomposition required for a hybrid DSA-MP process. Two schemes of such a hybrid process were studied. Optimal ILP formulation for the problem was presented for both schemes. Then we proposed efficient heuristic algorithms to solve the same problem, on the full-chip level, for each of the two schemes. The results of the heuristics are benchmarked against the ILP results. In our future work, we will generalize to other grouping structures that can be enabled by using EUVL to print the guiding templates.

VIII. Acknowledgments

This work was partly supported by IMPACT+ center (http://impact.ee.ucla.edu).

References

[1] P. A. Kearney, O. Wood, E. Hendrickx, G. McIntyre, S. Inoue, F. Goodwin, S. Wurm, J. van Schoot, and W. Kaiser, "Driving the industry towards a consensus on high numerical aperture (high-na) extreme ultraviolet (euv)," in *SPIE Proceedings Extreme Ultraviolet (EUV) Lithography V*, vol. 9048, 2014.

[2] Y. Ma, J. A. Torres, G. Fenger, Y. Granik, J. Ryckaert, G. Vanderberghe, J. Bekaert, and J. Word, "Challenges and opportunities in applying grapho-epitaxy dsa lithography to metal cut and contact/via applications," in *European Mask and Lithography Conference*. SPIE, 2014, pp. 92 310T–92 310T.

[3] Y. Seino, H. Yonemitsu, H. Sato, M. Kanno, H. Kato, K. Kobayashi, A. Kawanishi, T. Azuma, M. Muramatsu, S. Nagahara *et al.*, "Contact hole shrink process using graphoepitaxial directed self-assembly lithography," *Journal of Micro/Nanolithography, MEMS, and MOEMS*, vol. 12, no. 3, 2013.

[4] L.-W. Chang, X. Bao, C. Bencher, and H.-S. Wong, "Experimental demonstration of aperiodic patterns of directed self-assembly by block copolymer lithography for random logic circuit layout," in *Electron Devices Meeting (IEDM), IEEE International*, 2010.

[5] L.-S. Wan, P. A. Rincon Delgadillo, R. Gronheid, and P. F. Nealey, "Directed self-assembly of ternary blends of block copolymer and homopolymers on chemical patterns," *Journal of Vacuum Science Technology B: Microelectronics and Nanometer Structures*, vol. 31, no. 6, pp. 06F301–06F301–6, Nov. 2013.

[6] R. Ruiz, H. Kang, F. A. Detcheverry, E. Dobisz, D. S. Kercher, T. R. Albrecht, J. J. de Pablo, and P. F. Nealey, "Density multiplication and improved lithography by directed block copolymer assembly," *Science*, vol. 321, no. 5891, pp. 936–939, 2008.

[7] D. Z. Pan, L. Liebmann, B. Yu, X. Xu, and Y. Lin, "Pushing multiple patterning in sub-10nm: are we ready?" in *Proceedings of the 52nd Annual Design Automation Conference*. ACM, 2015, p. 197.

[8] A. B. Kahng, C.-H. Park, X. Xu, and H. Yao, "Layout decomposition for double patterning lithography," in *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press, 2008, pp. 465–472.

TABLE XI: Results on Complete Layouts with Sparse Rectangular Vias with QP, for the $Partial_DSA$ scheme. p_s is percentage of contacts assigned to special masks.

	Partial_MCM_H			Partial_Trivial_H			Partial_GP_MP		Partial_MP_GP		QP only
	Viol.	Time(s)	p_s (%)	Viol.	Time(s)	p_s (%)	Viol.	Time(s)	Viol.	Time(s)	Viol.
AES_srv	13	90	45	1	90	45	164	4	195	3	33
CortexM0_srv	13	88	45	2	94	45	132	9	162	6	28
LEON3_srv	52	797	35	19	2581	34	1650	11	2320	9	119
MIPS_srv	5	105	42	4	156	41	449	4	543	3	21
Comparison	3.19	0.37	1.01	1.00	1.00	1.00	92.12	0.01	123.85	0.01	7.73

[9] B. Yu, K. Yuan, B. Zhang, D. Ding, and D. Pan, "Layout decomposition for triple patterning lithography," in *Proc. ICCAD*, 2011.

[10] W. Zhao, H. Yao, Y. Cai, S. Sinha, and C. Chiang, "Fast and scalable parallel layout decomposition in double patterning lithography," *Integration, the VLSI Journal*, vol. 47, no. 2, pp. 175–183, 2014.

[11] H. Zhang, Y. Du, M. D. Wong, and R. Topaloglu, "Self-aligned double patterning decomposition for overlay minimization and hot spot detection," in *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*. IEEE, 2011, pp. 71–76.

[12] B. Xu, R. Piñol, M. Nono-Djamen, S. Pensec, P. Keller, P.-A. Albouy, D. Lévy, and M.-H. Li, "Self-assembly of liquid crystal block copolymer peg-b-smectic polymer in pure state and in dilute aqueous solution," *Faraday discussions*, vol. 143, 2009.

[13] M. Kim, E. Han, D. P. Sweat, and P. Gopalan, "Interplay of surface chemical composition and film thickness on graphoepitaxial assembly of asymmetric block copolymers," *Soft Matter*, vol. 9, no. 26, pp. 6135–6141, 2013.

[14] N. Jarnagin, "High x block copolymers for sub 20 nm pitch patterning: Synthesis, solvent annealing, directed self assembly, and selective block removal," Ph.D. dissertation, Geogia Institute of Technology, Dec. 2013.

[15] H. Yi, X.-Y. Bao, R. Tiberio, and H.-S. P. Wong, "Design strategy of small topographical guiding templates for sub-15nm integrated circuits contact hole patterns using block copolymer directed self assembly," vol. 8680, 2013.

[16] X.-Y. Bao, H. Yi, C. Bencher, L.-W. Chang, H. Dai, Y. Chen, P.-T. Chen, and H.-S. Wong, "SRAM, NAND, DRAM contact hole patterning using block copolymer directed self-assembly guided by small topographical templates," in *IEEE International Electron Devices Meeting*, 2011.

[17] C. T. Black, R. Ruiz, G. Breyta, J. Y. Cheng, M. E. Colburn, K. W. Guarini, H.-C. Kim, and Y. Zhang, "Polymer self assembly in semiconductor microelectronics," *IBM Journal of Research and Development*, vol. 51, no. 5, pp. 605–633, 2007.

[18] H. Kang, G. S. Craig, E. Han, P. Gopalan, and P. F. Nealey, "Degree of perfection and pattern uniformity in the directed assembly of cylinder-forming block copolymer on chemically patterned surfaces," *Macromolecules*, vol. 45, no. 1, pp. 159–164, 2011.

[19] D. Sundrani, S. Darling, and S. Sibener, "Hierarchical assembly and compliance of aligned nanoscale polymer cylinders in confinement," *Langmuir*, vol. 20, no. 12, pp. 5091–5099, 2004.

[20] Y. Du, D. Guo, M. D. F. Wong, H. Yi, H.-S. P. Wong, H. Zhang, and Q. Ma, "Block copolymer directed self-assembly (DSA) aware contact layer optimization for 10 nm 1d standard cell library," in *Proc. ICCAD*, ser. ICCAD '13. IEEE Press, 2013.

[21] Y. Badr, J. A. Torres, and P. Gupta, "Incorporating dsa in multipatterning semiconductor manufacturing technologies," in *SPIE Advanced Lithography*, 2015.

[22] J.-M. Brunet. (2013) Semiconductor engineering- "Let's all meet at the via bar". [Online]. Available: http://semiengineering.com/lets-meet-bar/

[23] J. Bekaert, J. Doise, R. Gronheid, J. Ryckaert, G. Vandenberghe, G. Fenger, Y. J. Her, and Y. Cao, "N7 logic via patterning using templated DSA: implementation aspects," in *Photomask Japan 2015*, N. Yoshioka, Ed. SPIE, jul 2015, p. 965804.

[24] Y. Ma, J. Lei, J. Torres, L. Hong, J. Word, G. Fenger, A. Tritchkov, G. Lippincott, R. Gupta, N. Lafferty *et al.*, "Directed self-assembly (dsa) grapho-epitaxy template generation with immersion lithography," in *Advanced Lithography*. SPIE, 2015, pp. 942 306–942 306.

[25] J. Edmonds, "Paths, trees, and flowers," *Canadian Journal of mathematics*, vol. 17, no. 3, pp. 449–467, 1965.

[26] A. W. Appel, *Modern compiler implementation in C*. Cambridge university press, 2004.

[27] Ucla hoffman2 cluster. [Online]. Available: https://idre.ucla.edu/hoffman2

[28] http://www.opencores.org.

[29] H. A. Darwish, H. N. Shagar, Y. A. Badr, Y. H. Arafa, and A. G. Wassal, "A hashing mechanism for rule-based decomposition in double patterning photolithography," in *2010 International Conference on Microelectronics*. IEEE, 2010, pp. 363–366.

Yasmine Badr is currently a PhD candidate in the Electrical Engineering Department at the University of California, Los Angeles (UCLA). She received her BSc and MSc degrees in Computer Engineering from Cairo University. Her current research interests are in computational methods for Design and Technology Co-optimization.

J. Andres Torres holds a B.S. in Chemical Engineering from the National Autonomous University of Mexico, a M.S. in Chemical Engineering from UW-Madison and a PhD degree in Electrical Engineering from OHSU. By understanding manufacturing and design requirements and restrictions, he has been investigating the interactions between process and electronic design flows. He has been with Mentor Graphics Design-to-silicon division since 2001 working in the area of Resolution Enhancement Technologies, Design for Manufacturing, Chemical Metal Polish and Stress. He is currently the advanced RET Flow Architect in the Calibre group and he is interested in streamlining the electronic design process for sub 10nm technologies. He has authored over 60 papers accepted in peer reviewed conference and journals and holds ten US patents, one European Patent and one Taiwanese Patent with four more pending in the area of electronic manufacturing and electronic design.

Puneet Gupta is a faculty member of the Electrical Engineering Department at UCLA. He received the B.Tech degree in Electrical Engineering from Indian Institute of Technology, Delhi in 2000 and Ph.D. in 2007 from University of California, San Diego. He co-founded Blaze DFM Inc. (acquired by Tela Inc.) in 2004 and served as its product architect till 2007. He has authored over 130 papers, 16 U.S. patents, a book and a book chapter. He is a recipient of NSF CAREER award, ACM/SIGDA Outstanding New Faculty Award, SRC Inventor Recognition Award and IBM Faculty Award. Dr. Gupta's research has focused on building high-value bridges across application-architecture- implementation-fabrication interfaces for lowered cost and power, increased yield and improved predictability of integrated circuits and systems.