Journal of
Micro/Nanolithography,
MEMS, and MOEMS

Nanolithography.SPIEDigitalLibrary.org

Design method and algorithms for directed self-assembly aware via layout decomposition in sub-7 nm circuits

Ioannis Karageorgos
Julien Ryckaert
Roel Gronheid
Maryann C. Tung
H.-S. Philip Wong
Evangelos Karageorgos
Kris Croes
Joost Bekaert
Geert Vandenberghe
Michele Stucchi
Wim Dehaene

Ioannis Karageorgos, Julien Ryckaert, Roel Gronheid, Maryann C. Tung, H.-S. Philip Wong, Evangelos Karageorgos, Kris Croes, Joost Bekaert, Geert Vandenberghe, Michele Stucchi, Wim Dehaene, "Design method and algorithms for directed self-assembly aware via layout decomposition in sub-7 nm circuits," *J. Micro/Nanolith. MEMS MOEMS* **15**(4), 043506 (2016), doi: 10.1117/1.JMM.15.4.043506.

Design method and algorithms for directed self-assembly aware via layout decomposition in sub-7 nm circuits

Ioannis Karageorgos,[a,b,*] Julien Ryckaert,[a] Roel Gronheid,[a] Maryann C. Tung,[c] H.-S. Philip Wong,[c] Evangelos Karageorgos,[d] Kris Croes,[a] Joost Bekaert,[a] Geert Vandenberghe,[a] Michele Stucchi,[a] and Wim Dehaene[a,b]
[a]imec, Kapeldreef 75, Leuven B-3001, Belgium
[b]KU Leuven, Department of Electrical Engineering (ESAT), Kasteelpark Arenberg 10, Leuven B-3001, Belgium
[c]Stanford University, Department of Electrical Engineering, 420 Via Palou, Stanford, California 94305, United States
[d]University of Athens, Department of Informatics and Telecommunications, Panepistimiopolis, Athens 15784, Greece

Abstract. Major advancements in the directed self-assembly (DSA) of block copolymers have shown the technique's strong potential for via layer patterning in advanced technology nodes. Molecular scale pattern precision along with low cost processing promotes DSA technology as a great candidate for complementing conventional photolithography. Our studies show that decomposition of via layers with 193-nm immersion lithography in realistic circuits below the 7-nm node would require a prohibitive number of multiple patterning steps. The grouping of vias through templated DSA can resolve local conflicts in high density areas, limiting the number of required masks, and thus cutting a great deal of the associated costs. A design method for DSA via patterning in sub-7-nm nodes is discussed. We present options to expand the list of usable DSA templates and we formulate cost functions and algorithms for the optimal DSA-aware via layout decomposition. The proposed method works *a posteriori*, after place-and-route, allowing for fast practical implementation. We tested this method on a fully routed 32-bit processor designed for sub-7 nm technology nodes. Our results demonstrate a reduction of up to four lithography masks when compared to conventional non-DSA-aware decomposition. [DOI: 10.1117/1.JMM.15.4.043506]

Keywords: directed self assembly; via patterning; design; cost function; grouping algorithms; alphabet.

Paper 16142P received Sep. 18, 2016; accepted for publication Oct. 13, 2016; published online Nov. 7, 2016.

1 Introduction

One of the main factors driving the growth of the semiconductor industry is the ability to print ever smaller features while keeping the cost at a minimum. In order to keep pace with Gordon Moore's famous semiconductor growth rate forecast,[1] lithographic process, the backbone of integrated circuit fabrication has seen significant advancements. However, as device dimensions keep shrinking, optical lithography is now facing some serious challenges in sustaining the required cost-effectiveness and overlay accuracy. As a consequence, the need for alternative lithography solutions is greater than ever.

Directed self-assembly (DSA) of block copolymers (BCPs) has emerged as a low-cost, high-throughput technique for extending the resolution of optical lithography. Although BCPs are used in several applications, ranging from automotive tires to drug delivery systems,[2] it is only recently that they have been extensively studied for use in the semiconductor industry. BCPs are formed when two or three monomers cluster together and form a chain of blocks. When the BCP is confined by physical guiding templates, its natural symmetry is altered and some aperiodic patterns can be formed inside the templates. Controlling the shape and position of the guiding templates, BCP materials self-assemble to form densely packed features with uniform dimensions and shapes, such as spheres, cylinders, or lamellae, in ordered arrays at the scale of 3 to 50 nm.[3–5] These nanostructures are of great interest in very-large-scale integration (VLSI) design. DSA can be regarded as a connection point between bottom-up self-assembly with top-down conventional photolithography design.[6,7]

There are several processing approaches for the utilization of self-assembled BCPs on nanolithography.[8] For the via application, one of the most straightforward approaches is the graphoepitaxy flow using cylindrical phase BCP materials and templated confinement.[9,10] In graphoepitaxy, an artificial topographic surface pattern is applied to control the orientation of crystal growth in thin films.[11,12] The self-assembly of a BCP thin film is guided through the topographically patterned substrate, creating well-aligned structures of BCP microdomains.[4] Using templated confinement, small clusters of closely packed BCP holes can be positioned accurately, allowing dense vias to be placed together on the same mask. The grouping of vias in the high-conflict areas allows the via layout to be printed with fewer multiple patterning (MP) steps, effectively cutting down a great deal of the associated costs, as well as limiting the performance impact of MP due to variability.[13,14]

In recent years, there has been significant progress in perfecting graphoepitaxial DSA from a processing standpoint. Collaborative efforts from academia and industry have optimized the DSA process for 300 mm fab-compatible implementation. Successive advancements in material development, defect control, and pattern transfer accuracy promoted DSA to a very promising approach for the patterning of vias in sub-7-nm technology nodes.[8–10,15–19] However,

*Address all correspondence to: Ioannis Karageorgos, E-mail: ioannis.karageorgos@imec.be

1932-5150/2016/$25.00

despite this great progress in the DSA process, a design approach for DSA-aware via decomposition is still far from practical implementation. Recent studies on the design domain present some promising solutions for tackling the problem of grouping "randomly" placed vias with finite template shapes and propose algorithms to optimize the grouping process.[20–22] These solutions, however, require a disruption in the electronic design automation (EDA) flow in terms of either the development of new place-and-route (PnR) tools or the adaptation of the current tools for a whole new set of algorithms. Modern commercial PnR tools have reached a certain maturity level and they are highly optimized to work in a conventional, non-DSA-aware, style. Such a change would require a sufficient amount of time for the tools to mature enough so that the benefits of using DSA would not be negated by the suboptimal PnR process. DSA-aware routing solutions introduce an area penalty, which also results in a power and performance penalty. Furthermore, these studies are based on small theoretical circuits, failing to provide an insight on the impact on large, complex, more realistic circuits.

In this article, we propose a method for DSA-aware via decomposition that minimizes the number of required masks while also taking into account the constraints of the DSA process. Rather than DSA-aware routing, in our method we develop algorithms for the via grouping in the derived layer after the PnR process. Similar to the alphabet approach,[21] we define a set of available DSA via templates, or DSA letters, and we use these letters to group the vias and decompose the layer. The set of DSA letters, or the DSA alphabet, is defined based on the available process flows and can be changed accordingly when new process options become available. The alphabet is then further expanded with the utilization of multiple BCPs combined with MP (DSA + MP). For the via grouping optimization, we formulate a cost function that is based on process flow data and we calibrate the coefficients with our in-house experimental results.

Our method provides a solution *a posteriori*, in the sense that it is applied after the PnR process. The main benefit of such a solution is that it does not require a disruption in EDA flow, thus it can be implemented rapidly. In order to demonstrate that, we developed a parameterized tool that uses our algorithms and performs full layout decomposition based on the provided DSA alphabet, configuration, and target lithographic options and parameters (e.g., ArF immersion, EUV). We use our tool to perform via decompositions on a fully routed ARM® Cortex®-M0 processor based on the 7-nm node, which we also scale down to various metal grid geometries in the range between 7- and 3-nm nodes. Our results demonstrate a reduction of up to four MP steps when we test our method targeting ArF immersion lithography, limiting the number of lithography masks to a maximum of three even on the denser 3-nm node.

The rest of this article is organized as follows. In Sec. 2 the problem statement is outlined, where we present results from our conventional via decomposition study on a fully routed processor based on several sub-7-nm technology nodes. Section 3 introduces some basic background on DSA via patterning and describes the imec templated DSA flow and process assumptions. In Sec. 4, our design method, the algorithms, the cost function, and the block diagram of our tool are presented. Section 5 introduces the results of our method and concludes this article.

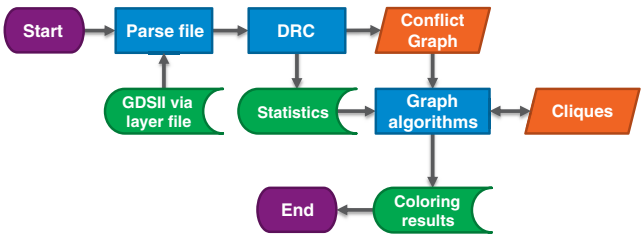

Fig. 1 Flowchart of the imec tool for via layer coloring.

2 Problem Definition

As VLSI technology proceeds beyond the 45-nm node, ArF immersion lithography, the workhorse of last decade's optical lithography has reached its effective limit. In the absence of a practical alternative,[23] the best feasible solution appeared to be the MP approach, where a dense circuit pattern can be partitioned into multiple separate exposures. This has proved a successful interim solution for several nodes, down to 10 nm. However, below the 10-nm node and as the number of required MP steps increases, the associated cost and variability penalty may render this solution prohibitive. Our assumption is that the maximum number of MP steps per metal layer should be limited to three, with an extreme maximum of four, in order for this option to remain cost effective. In order to quantify the increase of MP steps in realistic circuits below the 10-nm node, we studied the MP via decomposition of a fully routed ARM® Cortex®-M0 processor. This 32-bit processor is designed for the 7-nm technology node and is routed for three typical values of PnR utilization factor: 70%, 80%, and 90%. We define the utilization factor setting as the ratio of the die area to functional standard cells, in percentage. In this sense, 100% utilization factor would mean the highest possible density for the circuit, where all the area is occupied only by functional cells, without any dummy cells in-between (typically unrealistic). Then we scale each design to various dimensions down to 3-nm node, according to imec technology roadmap specifications, and we perform MP via decomposition, or coloring, for the most critical routing via layers. The coloring task is achieved by a custom tool for graph partitioning based on state-of-the-art coloring algorithms. The flowchart of this tool is illustrated in Fig. 1.

Initially, the tool generates the conflict graph from the input via layer and design rule check (DRC) and produces some graph statistics. Each via pair that violates DRC creates a graph edge. Then the first level of graph partitioning is performed and the maximal cliques are calculated using the recursive backtracking Bron–Kerbosch algorithm.[24] The coloring is generated using the Halldórsson and Lau algorithms,[25] followed by simulated annealing[26,27] in an iterative improvement loop, based on the maximum clique size. This iterative process starts with a chromatic number of one, which increases incrementally until an acceptable solution is found.

The results of our via coloring study for the first routing metal layers (metal2 up to metal5) of our test processor are shown in Fig. 2.

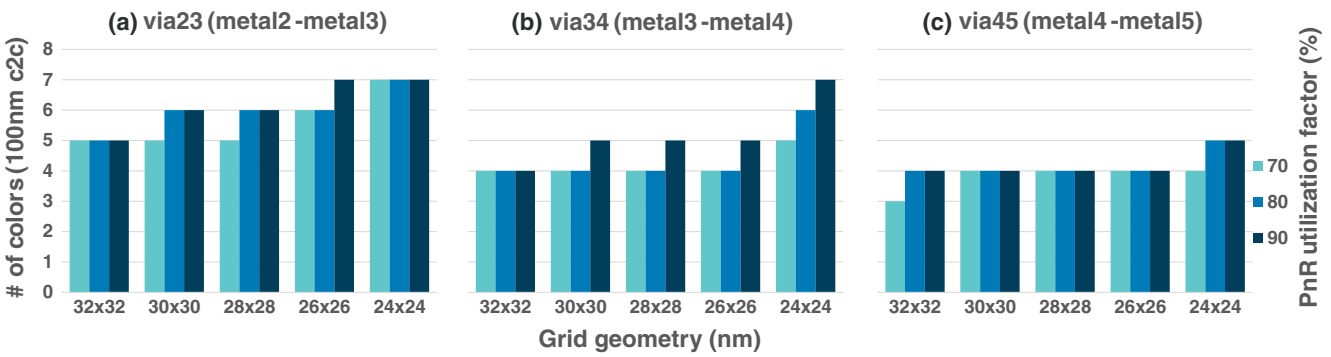

Fig. 2 ARM® Cortex®-M0 via coloring results. (a–c) represent a different via layer. For each layer, the minimum number of required colors is plotted on the *y* axis, across various metal grid geometries, which are shown as different column groups in the *x* axis, and for three different PnR utilization factor settings, illustrated with a different shade of cyan/blue.

The *y*-axis in each of the graphs of Fig. 2 shows the minimum number of colors (chromatic number) required to print the specific via layer. The column groups in the *x*-axis represent the different unidirectional metal grid geometries of our designs, which are based on the specifications of imec technology nodes in the range of 7 to 3 nm, and the different column shades represent the three PnR utilization factor settings used for the routing. Our results indicate that for the critical routing layers of a realistic circuit in sub-7-nm nodes, a minimum number of up to seven MP steps are required in order to print the vias, assuming ArF immersion lithography. This number is far above the maximum number of three MP steps per via layer we specified earlier. Even with the use of EUV lithography, a double patterning is required beyond the $26\ nm \times 26\ nm$ metal grid, since the minimum distance of diagonal vias is larger than the EUV resolution.[28,29] This increases the cost of an already expensive technology.

3 imec Templated Directed Self-Assembly Flow

3.1 *Process Description*

The imec templated DSA flow for vias has been described in detail in the past.[9,10,16,17] Since the process itself is not the main focus of this article, it will be discussed in brief and emphasis will be given only in the relevant design aspects. The imec DSA flow for vias is based on graphoepitaxy with cylindrical phase BCP materials. The schematic representation of this flow is shown in Fig. 3.

Starting with an Si or Si_3N_4 substrate, a typical trilayer stack consisting of 100 nm spin-on carbon, 30 nm spin-on glass, and 85 nm of negative tone development photoresist for ArF immersion are deposited on top using the Sokudo DUO coat and development system. The coating and development of the trilayer stack is done using the vendor's recommended settings for postapply bake, postexposure bake, and development. The prepattern templates are then patterned on the resist with an ASML NXT:1950i scanner and ArF immersion exposure. The exposure settings that are used include 1.2NA using annular illumination with $so = 0.8$, $si = 0.6$, and XY-polarization. Next, the templates are etched into the SoC/SoG stack and all resist material is removed.

At this stage, the template is, in principle, ready for BCP application. However, a process choice may be to strip the SoG layer by means of diluted HF and/or apply a brush layer to the template. As a final processing step, the BCP is being coated, typically resulting in partially conformal coating of the template topography. Annealing of the BCP results in phase separation of the blocks and thus the desired cylindrical hole patterns are formed. The cylindrical phase separation results in polymethyl methacrylate (PMMA) cylinders within the PS-block. Then a wet development process is used for the removal of these PMMA cores, leading to open DSA

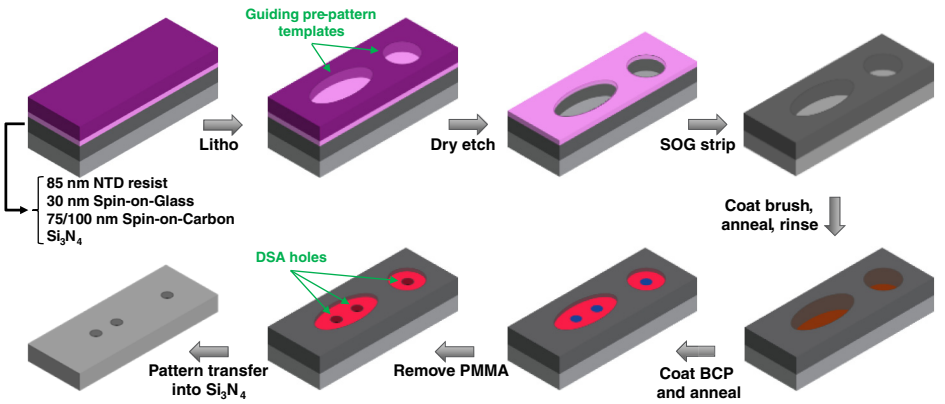

Fig. 3 Schematic overview of the imec templated DSA process flow.

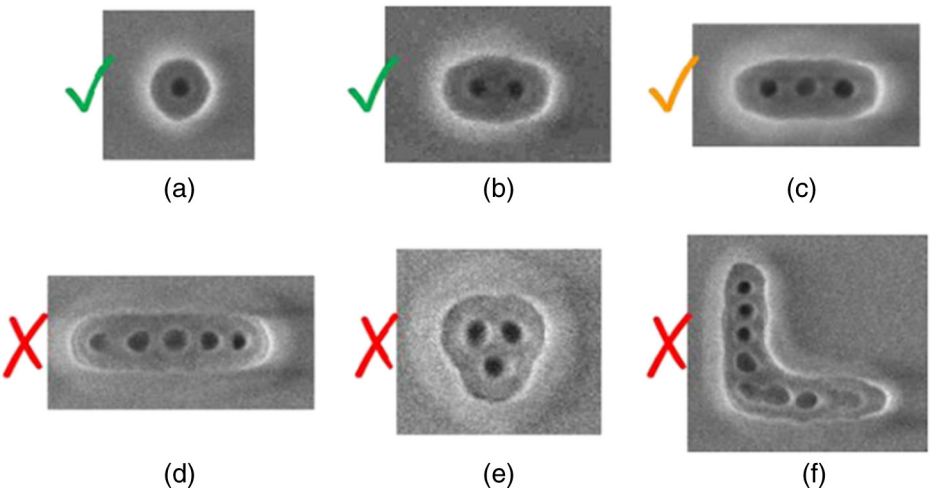

Fig. 4 SEM images of DSA hole formation for various template shapes. (a) The single-hole template which has the largest process window in our experimental data. (b) and (c) Double-hole and triple-hole one-dimensional templates with acceptable process windows. The larger one-dimensional templates, as in (d), as well as the complex two-dimensional templates in (e) and (f), show a negligible or nonexistent process window in our experiments.

holes. In our current process, we are testing two BCP materials with center-to-center natural pitches (L_0) of 37 and 48 nm, respectively.

3.2 *Pattern Restrictions*

When the shape and dimensions of the templates are commensurate with the BCP natural pitch, various configurations of DSA holes can be obtained within the template. Some typical examples of these configurations are shown in the scanning electron microscope (SEM) images of Fig. 4.

Our experimental results indicate that the larger templates lead to increasing variation in the CD, form, and placement of the DSA cylinder.[30] In addition, for more complex two-dimensional template shapes, like the ones shown in Figs. 4(e) and 4(f), variations in lithography (e.g., at corners) add further instability in the DSA formation. Therefore, starting from the capabilities of the templated DSA process in our hands, we currently restrict the pattern types considered for practical implementation of DSA to "singlets," "doublets," and at most "linear triplets" (referred to as just "triplets" for the rest of this article).

4 Design Method

In order to tackle the via decomposition problem, as described in Sec. 2, we introduce a method that combines 193i, MP, and templated DSA to drastically reduce the cost per via layer, which can be applied post-PnR, allowing fast practical implementation. The proposed method can be seen, on one hand, as an alternative to the much anticipated EUV lithography, but on the other hand, it can be also seen as a booster for EUV in the future, since it could reduce the number of masks when MP will eventually be required for EUV as well.

4.1 *Combine Block Copolymer Materials*

One of the challenges for the practical application of templated DSA is the lack of flexibility in the attainable distance between the cylindrical hole patterns, or the hole pitch, with respect to the pitch variation in the actual vias of a circuit. The pitch of the target vias has to be commensurate with the L_0 of the BCP. The pitch of the derived DSA holes can only vary slightly with the "stretching" or "compression" of the template.[31] As a result, there is a limited number of templated DSA letters that can be matched with the via patterns after the PnR.

In order to address this limitation and increase the number of available DSA letters, we suggest the use of multiple BCP materials in an MP flow. Taking advantage of the different process flow for each MP step, we could also utilize a different DSA flow that uses another BCP material. We assume that with the proper process optimization, any variability effect induced by the alternation of DSA process flows can be adequately limited. Combining BCP materials of different L_0 would increase the effective DSA hole pitch range, resulting in more DSA letters available for via grouping. As seen in the illustration of Fig. 5, one specific BCP material might successfully form holes in a diagonal via arrangement with a distance of ± 1, $\pm 1(x, y)$ grid points [Fig. 5(a)], but fail for the vertical arrangement with a distance of 0, ± 2 grid points [Fig. 5(b)]. Another BCP material could work for the 0, ± 2 grid points via arrangement [Fig. 5(d)], but fail for the diagonal vias [Fig. 5(c)]. Then a combination of both BCPs, in an MP flow, allows both via arrangements to be grouped by a DSA letter [Fig. 5(e)].

The careful choice of BCP materials, which should be based on the target technology dimensions, can greatly increase the DSA grouping flexibility.

4.2 *Templated Directed Self-Assembly Alphabets*

After specifying the BCP materials in use and the process flows, the next step is the definition of the feasible DSA letters that are available for via grouping. Our DSA alphabets are derived from experimental data based on a set of rules and a couple of assumptions. One of our assumptions is that the metals for the interconnects are unidirectional and are always placed on a grid. As a consequence, a fixed grid is created for the via placement and in that sense any pair of vias can be separated on the x or y axis by a distance

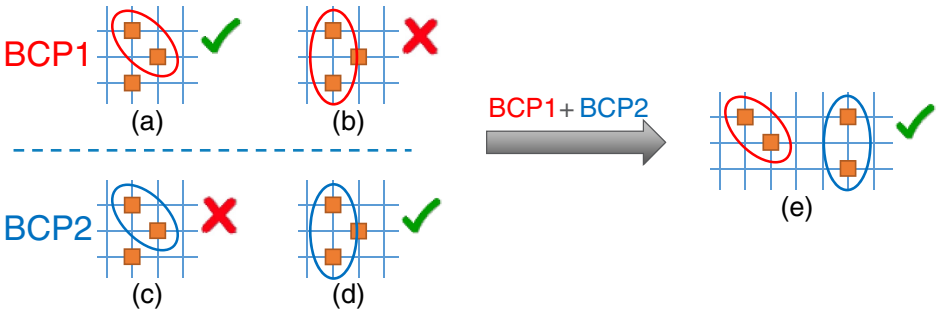

Fig. 5 Schematic illustration of the expansion of available DSA letters when combining BCP materials of different L_0. (a) Accurate hole formation with BCP1 when grouping diagonal vias (via pitch very close to L_0). (b) Inaccurate hole formation with BCP1 when grouping vias in horizontal/vertical arrangement (via pitch much larger than L_0). (c) Inaccurate hole formation with BCP2 when grouping diagonal vias (via pitch much smaller than L_0). (d) Accurate hole formation with BCP2 when grouping vias in horizontal/vertical arrangement (via pitch very close to L_0). (e) A combination of the two materials, in different process steps, allows accurate hole formation when grouping both via arrangements.

that is always a multiple of the grid pitch. Due to the unidirectionality of metals and the minimum clearance distance (metal end-to-end) that is needed in lithography, a design rule forbids the placement of two vias adjacent (on the immediate next grid point) on the x or y axis. The router makes sure that no connection can be put there. Instead, it will try to make the connection further away, at least two grid points apart (± 2, 0 or 0, ± 2 for x, y). However, a pair of vias can be adjacent in the diagonal direction, which is ± 1, $\pm 1(x, y)$ grid points away, since this does not create any violation. Thus, in a densely packed group of vias, the via arrangement tends to resemble a checkerboard type of pattern where the minimum feature distance is the distance of the diagonally adjacent vias ($\sqrt{\text{grid}_x^2 + \text{grid}_y^2}$). Another assumption is that although our experimental results involve only horizontal or vertical templates for "doublets" and "triplets", we assume that templates of any orientation on the layer can be achieved.

Taking into account the above rules and restrictions, we define a library of DSA alphabets, each one coupled with a specific technology. In this study, two BCP materials are considered, one with an L_0 of 37 nm and one with an L_0 of 48 nm. A schematic representation of our DSA alphabets across various metal grid geometries is shown in Fig. 6.

In Fig. 6 we show a set of feasible DSA letters, with each color representing one of the BCP materials in use. These letters are derived from the commensuration of the possible via arrangements on a specific grid, with the pitch range of each BCP material, as shown in the top left section of the figure. The latter represents the stretching/compression range of the template around the natural pitch of the material while still having a successful hole formation.

For better visibility we show the letters in only one orientation, although they can have any orientation on the grid (e.g., 90 deg rotated, mirrored on one axis). Additionally, wherever we show a "triplet," it is implied that a "doublet" of the same form is also available. However, this does not apply vice versa. The reason is that in our process the "triplet" has a narrower process window than the "doublet," which translates to a narrower hole pitch range. In this regard, if a "triplet" of a specific pitch value is achievable, then an equivalent "doublet" is certainly also achievable; however, the opposite may not apply.

For each alphabet, we set n classes of letters, where n is the number of available BCP materials; in our case, two

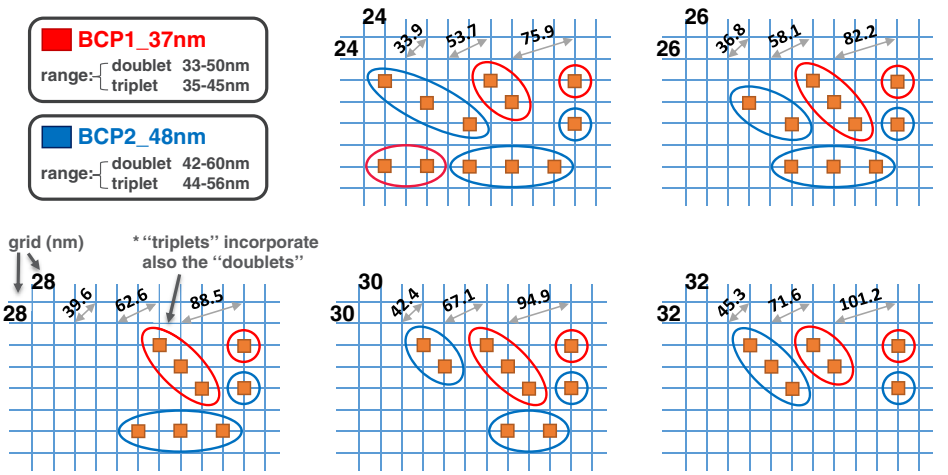

Fig. 6 Overview of our DSA alphabets definition for various technology nodes (metal grid pitch) using two BCP materials (top left region of the figure).

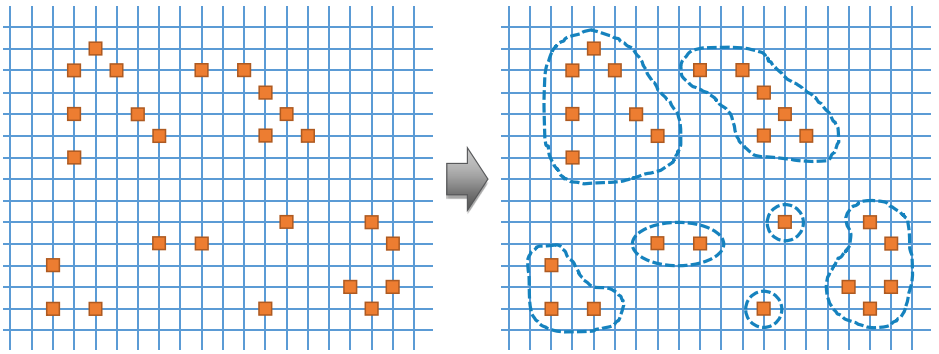

Fig. 7 Schematic illustration of the layout split into noncolor-conflicting via clusters.

classes: class A and class B. Then we set seven different subclasses of letters: (a) the "singlet," (b) the "rectilinear doublet" (always aligned with the x or y axis), (c) the "rectilinear triplet," (d) the "diagonal doublet" (vias separated by ± 1, ± 1 grid points), (e) the "diagonal triplet," (f) the "skewed doublet" (vias separated by ± 2, ± 1 or ± 1, ± 2 grid points), and (g) the "skewed triplet." Each subclass (except for the "singlet") includes all the different versions of the relevant letter in any orientation, with each variant assigned a different ID.

With the classification in place, we compose our alphabet configurations, which are basically all the combinations of the basic classes for the relevant range of colors (MP steps) we target. For example, for one color (single patterning) there are two alphabet configurations: A and B. This means that either one of the two BCP materials is going to be used and we can test both to see which one will be more suitable. For two colors, the possible alphabet configurations are AA, AB, and BB ($BA = AB$). The AA configuration means that only the first material will be used for both patterning steps, AB means that both BCP materials can be used, one for each of the patterning steps, and BB means that only the second BCP material will be used for both patterning steps. For three colors, the alphabet configurations are AAA, AAB, ABB, and BBB. The following general equation is derived for the calculation of the number of alphabet configurations with n number of classes and r number of colors

$$\binom{n}{r} = \frac{(n+r-1)!}{r!(n-1)!}.$$

In the present study, for $n = 2$, the number of configurations is $r + 1$.

4.3 *Layout Split*

Graph partitioning and coloring has been shown to be an NP-hard problem[32–34] (unless $P = \mathrm{NP}$). In a large realistic circuit with tens or hundreds of thousands of vias, it would be nearly impossible to find a good solution in finite time. As a first optimization step, we propose the split of the layout into smaller noncolor-conflicting via clusters. All vias that belong to the same cluster are separated from any other via by a distance that is larger than the lithography resolution; in our experiments we assume a minimum distance of 100 nm for immersion lithography resolution. This way, the via layer is divided into small via clusters that can be colored separately since whatever coloring scheme is used in one cluster does not have an impact on the coloring scheme of any other. Likewise, from the graph point of view, the clustering task is a type of unbalanced graph partitioning into components (vertex, edge, or set of edges) of minimum size, with zero edges running between each component. A schematic illustration of this process is shown in Fig. 7.

Computation of the graph components can be done in linear time, thus optimizing a lot the grouping and coloring processes since the number of vertices can be significantly reduced; these components can be grouped and colored separately, one at a time.

4.4 *Grouping and Coloring*

4.4.1 *Cost function*

In order to group the vias in a way that can be matched with a DSA letter, we have to somehow "scan" each cluster with the letter pattern and identify the possible matches. Each cluster's via pattern, however, can be matched in multiple ways with various combinations of letters. An example of different grouping solutions is shown in Fig. 8.

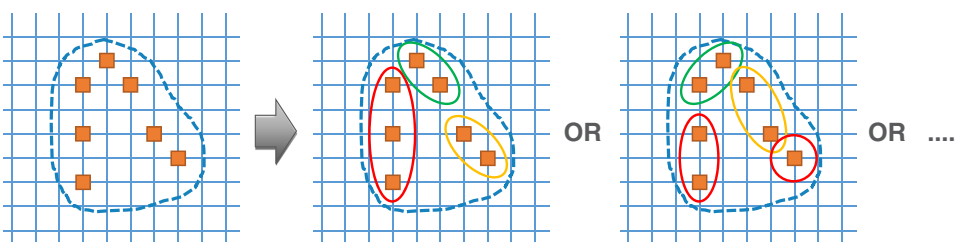

Fig. 8 Illustration of different grouping solutions applied to a via cluster.

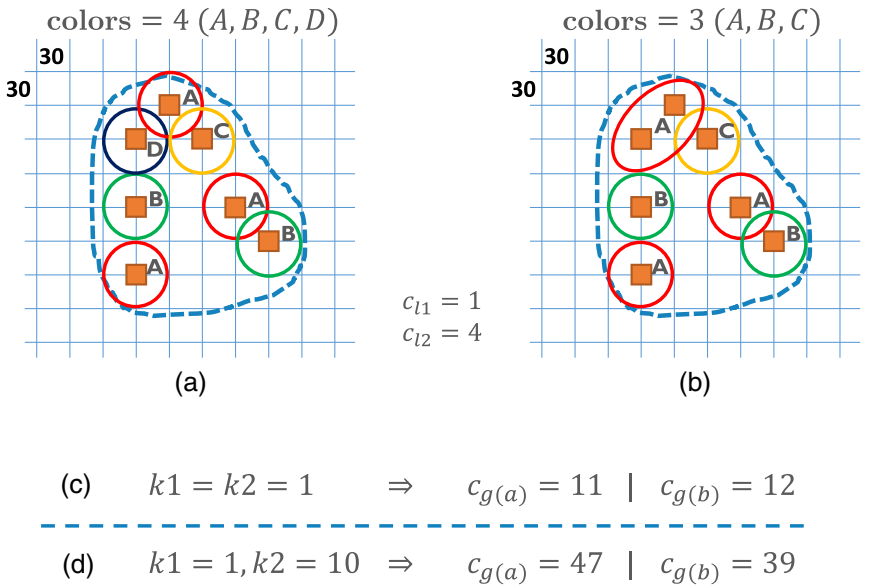

Fig. 9 Examples of grouping cost values for different grouping solutions, (a) and (b), and their respective grouping cost values when (c): $k_1 = k_2$ and (d): $k_2 = 10k_1$. The letter cost of the "singlet" (c_{l1}) is assumed to be one unit (one unit per via) and the cost of the "doublet" is four units (two units per via).

In order to qualify each of the possible grouping combinations and optimize the solution, we formulate a cost function. An initial cost value is assigned to each letter, which is derived from experimental data and is related mainly to the process window for the successful creation of the target holes after PMMA removal. Starting with a reference cost of one unit for the "singlet," it is up to the process engineers to estimate the cost values for the rest of the letters, based on their experimental results. In this study, we start with the assumption of a linear relation among the three basic categories of our letters: the "singlet," the "doublet," and the "triplet." We assign a cost of two units for the "doublet" and three units for the "triplet." We also assume an equal cost for all versions of "doublets" and "triplets," which are the letters in all the different orientations. Our experimental data so far include only rectilinear templates for the long letters, thus we cannot have a safe estimation of the cost variations in different orientations, if they do exist.

The total cost of each grouping solution can be estimated as the sum of two components: (a) the cost of all the individual letters used for this grouping and (b) the number of required colors. The following cost function is formulated for the estimation of any grouping cost c_g:

$$c_g = k_1 \times \sum_{l \in g} c_l + k_2 \times r,$$

where c_l denotes the letter cost for each letter l multiplied by the number of vias of that letter [$c_l = (\text{letter cost units}) \times (\text{number of vias})$], r denotes the number of colors used in each grouping g and k_1, k_2 are weight constants specifying the relative importance between the letter cost summation, which translates to process optimization, and the target number of colors.

An example of the total grouping cost values for different grouping solutions applied to the via cluster of Fig. 8 is shown in the illustration of Fig. 9. The grouping solution in Fig. 9(a) uses four colors and the solution in Fig. 9(b) uses three colors. Assuming a cost of one unit for the "singlet": $c_{l1} = 1$ and four units for the "doublet" (two units per via $\times$ 2 vias): $c_{l2} = 4$, if we set $k_1 = k_2$, then the optimal grouping is the one of Fig. 9(a), since it has the lowest grouping cost value $\rightarrow$ Fig. 9(c): $c_{g(a)} < c_{g(b)}$. However, if we increase the weight of k_2 by 10 times the weight of k_1, which means that we are concerned more about the number of colors rather than the process window, then the grouping of Fig. 9(b) is the most favorable one $\rightarrow$ Fig. 9(d): $c_{g(a)} > c_{g(b)}$.

4.4.2 *Grouping and coloring algorithms*

In order to find an appropriate grouping solution, which will be referred to as solving the cluster, we have developed a process that takes the set of all possible letter matches as an input and yields a specific subset of them that covers most, if not all, vias of the cluster only once and has a minimal aggregate cost. The matches have an inherent connectivity based on the color of their letter and their position. Two matches are overlapping if at least one via is covered by both of them. Two matches are neighbors if the distance between them is below the conflict threshold. Two matches are conflicting if they are neighbors and have the same color. Otherwise, they are nonconflicting. Two matches are nonconflicting neighbors if they are neighbors, nonconflicting, and not overlapping. In order to solve the cluster, we propose two different algorithmic approaches based on two different data organizational structures. In the solution tree approach, we model all possible match sets as nodes on a tree. Any match set is defined as a tree node. Every tree node can generate a child node by removing a match from its set. This means that any tree node that contains n matches will have n children, each of which will have $n-1$ matches. The total number of descendants (d) of a node will be $d(n) = n \times [d(n-1) + 1]$, with $d(1) = 0$, or alternatively $d(n) = \lfloor (e-1) \times n! \rfloor - 1$.[35] The cost of a tree node is the aggregate cost of all the matches in its set. A root tree

node along with all its descendants defines a solution tree. Also, a node that fails to cover all the vias of the cluster is invalid and its children can be pruned from the tree. A tree node whose matches are nonconflicting and cover all the vias of the cluster only once is a solution node, while the solution node that has the lowest cost is the optimal solution node.

Another algorithmic approach models all the matches as a graph, called the match graph. The nodes of this graph are the matches themselves, while any two matches that are neighbors are connected by an edge. By traversing the graph and selecting matches that are nonconflicting with each other, we can yield a solution set.

Using the solution tree and match graph algorithmic approaches, we define three different algorithms that attempt to solve a cluster. These algorithms can be combined to create a process that solves any cluster efficiently with a high success rate and a minimal cost.

a. The tree solving algorithm traverses the solution tree to find a solution node with minimal cost. Performing breadth-first search (BFS) on the solution tree, all the solution nodes can be exhaustively found along with the optimal solution node and, subsequently, the optimal grouping solution. This is an exhaustive search over the solution tree and will always find the optimal solution. It only fails to find a solution if there is none, and in this case, it finds the best possible fit. The combinatorial complexity, combined with a large search space, makes this algorithm practical only for clusters with a small number of vias. Alternatively, a heuristic depth-first search can be performed on the tree, stopping on the first available solution node. This can yield a solution significantly quicker, but the solution can be suboptimal.
b. The path solving algorithm traverses the match graph, creating a variant of a simple path whose matches are nonconflicting and cover most, if not all, of the vias in the cluster. In each step, there is a transition from a match to one of its nonconflicting neighbors. The selected neighbor must not conflict with any of the matches already in the path. The path depends on the specific neighbor chosen from a node's transition to another. If it fails to cover all the vias in the cluster, it recursively backtracks to choose different neighbors of previous matches, resulting in a different path. The graph traversal itself is performed in linear time; however, the involvement of recursive backtracking implicates that unless a viable solution is quickly found, the search has combinatorial complexity. Although this algorithm can have nonpolynomial complexity, the search space is significantly smaller than the search space of the solution tree. Most clusters that consist of less than $\sim$10 vias are solved with hardly any backtracks, almost instantaneously. The algorithm quickly converges to a solution, since the first path that is found covers almost all vias of the cluster. This approach is similar to the way humans intuitively try to solve the same problem; start from a match and try to add valid adjacent matches until the entire cluster is solved, while when a dead-end is encountered, undo some steps and try a slightly different path.
c. The generational solving algorithm traverses the match graph using a variant of graph growing. A generation is a set of nonconflicting matches. It has its own search front, consisting of all the neighboring matches at the boundary of the covered area. A generation can spawn a new generation by choosing matches from its search front to expand the match set. Successive generations will be spawned, until either no more matches can be included, or the last generation's match set covers all the vias of the cluster. The matches that are added to a generation's set during spawning depend on the ordering of the matches in the search front. If the algorithm fails to result in a generation that covers all the vias, it recursively backtracks and changes the ordering of the matches in previous generations' search fronts, resulting in different sets of matches for the next generations. The generational solving algorithm is the most scalable approach, yielding a solution for a cluster of 100+ vias in less than a second on commodity hardware. One of the side effects of this algorithm is that it cannot be exhaustive, since the differentiation of results is based on the order of the matches in the cluster, which is arbitrary.

The three algorithms are utilized in a number of successive steps in order to solve a particular cluster. Every step performs a faster algorithm than the previous one, but is expected to find a less optimal solution. If any of the steps yield a solution, no more steps will be attempted. The steps are as follows:

1. If the cluster is small enough, execute the tree solving algorithm with a time limit.
2. Try to find a solution using the path solving algorithm, configured to search for a low solution cost.
3. Try to find a solution using the path solving algorithm, configured to perform aggressive optimization and search for a low solution cost.
4. Try to find a solution using the path solving algorithm, configured to perform aggressive optimization and search for best fit.
5. Try to find a solution using the generational solving algorithm.
6. Randomize the order of the matches in the cluster and try to find a solution using the generational solving algorithm.
7. Repeat last step N times, where N is given in configuration.

If a solution is not found after all the steps, the process returns the set which covers the most vias, while having the lowest cost.

4.4.3 *Directed self-assembly aware via decomposition tool*

Putting everything together, we developed a tool for the optimal DSA-friendly decomposition of a via layer based

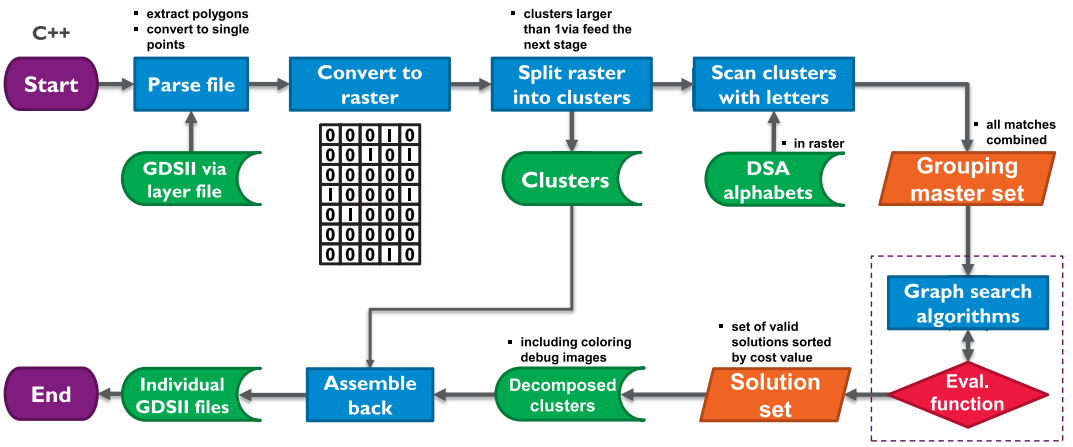

Fig. 10 Flowchart of our DSA-aware via layout decomposition tool.

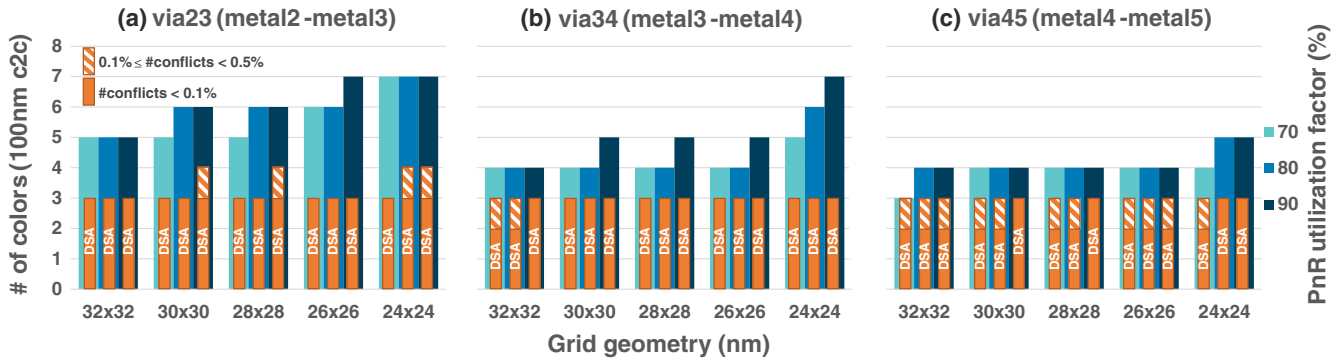

Fig. 11 DSA via decomposition results. Comparative column bar chart overlaid (orange columns) to the bar chart results of Fig. 2 (cyan/blue shaded columns).

on provided alphabets and configurations. The basic flowchart of our tool is shown in Fig. 10.

The tool inputs are the circuit layout file in GDSII format, the target via layer name, an alphabet file, and a configuration file. Initially the tool parses the GDSII file and extracts the polygons of the target via layer saving them in bBox format. Then the polygons are converted to single points with coordinates of the center of mass of each polygon. The actual via polygon is not relevant at this point, only their relative positions are important. After that, the tool performs the via clustering task using a BFS algorithm for connected components, optimized with space partitioning. The results are saved on disk as separate files. The clusters that are bigger than one-via enter the basic tool flow one by one where they are first rasterized (converted to matrices) and then they are processed using our grouping and coloring algorithms, as described in Sec. 4.4.2. The best solutions are saved on disk along with the total cost. The clusters are assembled back again, they are converted back to polygons and then converted and saved back to the individual GDSII files, one for each mask.

5 Results and Discussion

We used our tool to run a decomposition study for all versions of our benchmark ARM® Cortex®-M0 processor, as described in Sec. 2. The results of this study are presented in the comparative column bar chart of Fig. 11, overlaid to the non-DSA decomposition results of Fig. 2. The solid orange bars represent the minimum number of colors in each decomposition, when the remaining conflicts are less than 0.1% of the total number of vias. In these cases either there is no conflict at all or there are only a few conflicts here and there which we assume they can be easily resolved by a manual change of the layout. The orange bars which are complemented with a dashed bar indicate that for the solid part of the bar the remaining conflicts are limited between 0.1% and 0.5% of the total number of vias and an extra color (dashed bar) may be needed. For this range of number of conflicts, we assume that a manual resolve may be applicable for small to medium circuits; however, it might not be manageable for large circuits. After that level ($\geq 0.5\%$) we assume that a manual resolve is no longer possible, or at least worthwhile.

These results indicate that a reduction of up to four MP steps can be achieved in a realistic circuit below the 7-nm node, using the described method along with the considered DSA flow and materials. Even on the denser metal grid geometry of $24\ nm \times 24\ nm$ pitch, the number of MP steps is limited to a maximum of three in most of the cases. Moreover, this reduction of MP steps can be achieved after the PnR process, leaving the EDA flow intact, and thus allowing for fast adoption. The utilization of more BCP materials, which translates to the expansion of alphabets, can further increase the efficiency of this method.

Acknowledgments

This work was supported in part by the National Science Foundation (award number 1421292). MCT is additionally supported by a PhD Fellowship of the National Science Foundation.

References

1. G. E. Moore, "Progress in digital integrated electronics," in *Int. Electron Devices Meeting (IEDM '75) Technical Digest*, Vol. 21, pp. 11–13 (1975).
2. H. Yoshida and M. Takenaka, "1—Physics of block copolymers from bulk to thin films," in *Directed Self-Assembly of Block Co-Polymers for Nano-Manufacturing*, R. Gronheid and P. Nealey, Eds., Woodhead Publishing Series in Electronic and Optical Materials, pp. 3–26, Woodhead Publishing, Sawston, Cambridge (2015).
3. I. Hamley, *The Physics of Block Copolymers*, Oxford Science publications, Oxford University Press, Oxford (1998).
4. C. Park, J. Yoon, and E. L. Thomas, "Enabling nanotechnology with self assembled block copolymer patterns," *Polymer* **44**(22), 6725–6760 (2003).
5. M. P. Stoykovich et al., "Directed self-assembly of block copolymers for nanolithography: fabrication of isolated features and essential integrated circuit geometries," *ACS Nano* **1**(3), 168–175 (2007).
6. S. O. Kim et al., "Epitaxial self-assembly of block copolymers on lithographically defined nanopatterned substrates," *Nature* **424**(6947), 411–414 (2003).
7. C. T. Black et al., "Polymer self assembly in semiconductor microelectronics," *IBM J. Res. Dev.* **51**, 605–633 (2007).
8. J. K. Kim et al., "Functional nanomaterials based on block copolymer self-assembly," *Prog. Polym. Sci.* **35**(11), 1325–1349 (2010).
9. J. Bekaert et al., "Contact hole multiplication using grapho-epitaxy directed self-assembly: process choices, template optimization, and placement accuracy," *Proc. SPIE* **9231**, 92310R (2014).
10. R. Gronheid et al., "Process optimization of templated DSA flows," *Proc. SPIE* **9051**, 90510I (2014).
11. H. I. Smith and D. C. Flanders, "Oriented crystal growth on amorphous substrates using artificial surface-relief gratings," *Appl. Phys. Lett.* **32**(6), 349–350 (1978).
12. H. I. Smith et al., "Silicon-on-insulator by graphoepitaxy and zone-melting recrystallization of patterned films," *J. Cryst. Growth* **63**(3), 527–546 (1983).
13. K. Jeong, A. B. Kahng, and R. O. Topaloglu, "Assessing chip-level impact of double patterning lithography," in *11th Int. Symp. on Quality Electronic Design (ISQED '10)*, pp. 122–130 (2010).
14. I. Karageorgos et al., "Impact of interconnect multiple-patterning variability on SRAMs," in *Design, Automation Test in Europe Conf. Exhibition (DATE '15)*, pp. 609–612 (2015).
15. S.-J. Jeong et al., "Directed self-assembly of block copolymers for next generation nanolithography," *Mater. Today* **16**(12), 468–476 (2013).
16. R. Gronheid et al., "Implementation of templated DSA for via layer patterning at the 7 nm node," *Proc. SPIE* **9423**, 942305 (2015).
17. J. Doise et al., "Implementation of surface energy modification in graphoepitaxy directed self-assembly for hole multiplication," *J. Vac. Sci. Technol. B* **33**(6), 06F301 (2015).
18. J. Doise et al., "Influence of template fill in graphoepitaxy DSA," *Proc. SPIE* **9779**, 97791G (2016).
19. R. Gronheid et al., "EUV patterned templates with grapho-epitaxy DSA at the N5/N7 logic nodes," *Proc. SPIE* **9776**, 97761W (2016).
20. H. Yi et al., "Contact-hole patterning for random logic circuits using block copolymer directed self-assembly," *Proc. SPIE* **8323**, 83230W (2012).
21. Y. Du et al., "Block copolymer directed self-assembly (DSA) aware contact layer optimization for 10 nm 1d standard cell library," in *IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD '13)*, pp. 186–193 (2013).
22. Y. Du et al., "DSA-aware detailed routing for via layer optimization," *Proc. SPIE* **9049**, 90492J (2014).
23. R. S. Ghaida and P. Gupta, "Role of design in multiple patterning: technology development, design enablement and process control," in *Design, Automation Test in Europe Conf. Exhibition (DATE '13)*, pp. 314–319 (2013).
24. C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," *Commun. ACM* **16**, 575–577 (1973).
25. M. M. Halldrsson and H. Lau, "Low-degree graph partitioning via local search with applications to constraint satisfaction, max cut, and coloring," *J. Graph Algorithms Appl.* **1**(3), 1–13 (1997).
26. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science* **220**(4598), 671–680 (1983).
27. V. Černý, "Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm," *J. Optim. Theory Appl.* **45**(1), 41–51 (1985).
28. A. Yen, "Advanced lithography," in *IEDM Short Course: Emerging CMOS Technology at 5 nm and Beyond*, p. 23 (2015).
29. C. Lin et al., "Feasibility study of grapho-epitaxy DSA for complementing EUV lithography beyond N10," in *1st Int. Symp. on DSA* (2015).
30. G. Fenger et al., "Calibration and application of a DSA compact model for graphoepitaxy hole processes using contour-based metrology," *Proc. SPIE* **9235**, 92351X (2014).
31. S. M. Nicaise, K. A. Tavakkoli, and K. K. Berggren, "8—self-assembly of block copolymers by graphoepitaxy," in *Directed Self-Assembly of Block Co-Polymers for Nano-Manufacturing*, R. Gronheid and P. Nealey, Eds., Woodhead Publishing Series in Electronic and Optical Materials, pp. 199–232, Woodhead Publishing, Sawston, Cambridge (2015).
32. K. Andreev and H. Räcke, "Balanced graph partitioning," in *Proc. of the Sixteenth Annual ACM Symp. on Parallelism in Algorithms and Architectures (SPAA '04)*, pp. 120–124, ACM, New York, NY (2004).
33. A. E. Feldmann and L. Foschini, "Balanced partitions of trees and applications," *Algorithmica* **71**(2), 354–376 (2015).
34. A. Buluç et al., "Recent advances in graph partitioning," CoRR, http://arxiv.org/abs/1311.3144 (2013).
35. D. E. Knuth, *The Art of Computer Programming. Combinatorial Algorithms, Part 1*, Vol. **4A**, Addison-Wesley Professional, Boston (2011).

Ioannis Karageorgos is a PhD student at imec, Belgium, affiliated with the Electrical Engineering Department of University of Leuven. He received his BS degree in electrical engineering from ASPETE, Athens in 2008 and his MS degree in microelectronics from the University of Athens in 2012. His current research interests include circuit design methods, computer science, and electronic design automation. He is a member of SPIE.

Biographies for the other authors are not available.