A Boolean Rule-Based Approach for Manufacturability-Aware Cell Routing

Jordi Cortadella, *Member, IEEE,* Jordi Petit, Sergio Gómez, and Francesc Moll, *Member, IEEE*

***Abstract*—An approach for cell routing using gridded design rules is proposed. It is technology-independent and parameterizable for different fabrics and design rules, including support for multiple-patterning lithography. The core contribution is a detailed-routing algorithm based on a Boolean formulation of the problem. The algorithm uses a novel encoding scheme, graph theory to support floating terminals, efficient heuristics to reduce the computational cost, and minimization of the number of unconnected pins in case the cell is unroutable. The versatility of the algorithm is demonstrated by routing single- and double-height cells. The efficiency is ascertained by synthesizing a library with 127 cells in about one hour and a half of CPU time. The layouts derived by the implemented tool have also been compared with the ones from a commercial library; thus, showing the competitiveness of the approach for gridded geometries.**

***Index Terms*—Cell generation, design for manufacturability, detailed routing, satisfiability.**

I. Introduction

THE LITHOGRAPHIC gap between the light wavelength and the actual feature sizes is having a prominent impact on the patterns used for layout generation in current manufacturing processes. Various resolution enhancement techniques are applied to obtain faithful printed feature sizes much smaller than the light wavelength [1]. The effectiveness of these techniques is limited, however, to certain geometrical configurations, restricting the set of allowed layout shapes. This fact has contributed to an enormous increase of layout design rules at each technology generation.

Litho-friendly layout techniques are vital for current and future technology nodes. These techniques exploit the use of 1-D features and gridded locations for the layout elements. However, these constraints complicate the design of cell libraries that must be efficient in area, performance and power. The good news is that litho-friendly layouts are more tractable by EDA tools.

Manuscript received May 16, 2013; revised October 1, 2013; accepted November 12, 2013. Date of current version February 14, 2014. This work was supported in part by Intel Corporation, in part by the Project FORMALISM under Grant CICYT TIN2007-66523, in part by the Generalitat de Catalunya under Grant ALBCOM-SGR 2009-2013, and in part by the Spanish Ministry of Economy and FEDER funds through the Project TEC2008-01856. This paper was recommended by Associate Editor C. C.-N. Chu.

J. Cortadella and J. Petit are with the Department of Software, Universitat Politècnica de Catalunya, Barcelona 08034, Spain (e-mail: jordi.cortadella@upc.edu; jpetit@lsi.upc.edu).

S. Gómez and F. Moll are with the Department of Electronic Engineering, Universitat Politècnica de Catalunya, Barcelona 08034, Spain (e-mail: sergio.gomez-fernandez@upc.edu; francesc.moll@upc.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2013.2292514

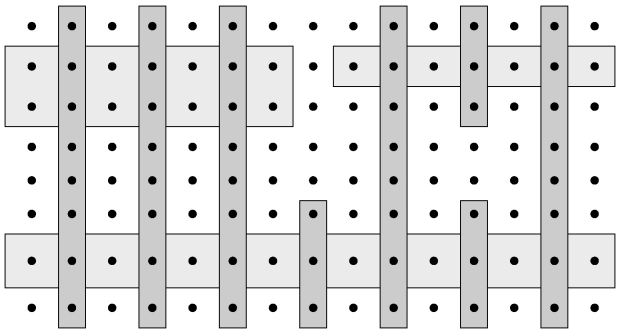


Fig. 1. Polysilicon and diffusion layers in a gridded layout.

A. Cell Synthesis

The synthesis of cell libraries has been one of the targets for design automation since long ago [2], [3]. With large feature dimensions, the layout elements could be placed with very simple design rules. Today, the lithography gap is imposing complex design rules that highly complicate automation.

Regularity has been recently introduced as a means to make design automation tractable. A clear example is the lithographer's dream patterns (LDP) [4] in which all layout elements are located on a virtual grid of evenly spaced points, ideally using a grid unit that is a fraction of the wavelength of the light sources.

A typical gridded layout for the FEOL layers of a standard cell is depicted in Fig. 1. The transistors are aligned in two diffusion strips (p and n), while polysilicon is laid out with vertical rectangles [5]. The diffusion strips can be isolated by connecting polysilicon gates to Vdd or Vss or by removing polysilicon and diffusion.

Different approaches have been proposed using the LDP paradigm, each of them with advantages and limitations. In [6] and [7], a methodology based on synthesizing small logic bricks was proposed. The approach in [7] was based on a Boolean formulation of the problem, inspired by the model proposed in [8] and extended to support technology-specific design rules. However, the complexity of the problem restricted the applicability to small cells.

The work in [5] evaluated the impact of using a regular layout fabric in an industrial flow. Different algorithms for design automation were proposed to synthesize a small subset of cells. In that case, the algorithms were specially customized for that fabric and for a specific set of design rules.

The synthesis of a standard cell is typically decomposed into two main tasks: transistor placement and routing. The first task calculates a linear arrangement of the p and n transistors along the diffusion strips [9], [10]. The second task find routes to connect transistors using the available metal layers. The goal of this paper is to propose an algorithmic approach for a generic cell router.

B. Generic Cell Router

Ideally, a cell router should be able to survive along the changing parameters by using technology-independent techniques. We next enumerate some desirable properties for a generic cell router.

1) The routing algorithm should be independent from the layout templates and the interconnect resources used for cell synthesis. In this way, the cell router can be parameterized with the set of resources available for each technology generation.
2) A set of parameterizable attributes should be allowed for every wire segment. For example, segments with different width (thin/thick) or assigned to different masks (for multiple patterning) are typically used in nanometric technologies.
3) The routing algorithm should be independent from the set of design rules. Design rules should be a parameter of the router and customizable according to the attributes assigned to the wire segments. For example, different rules might be applicable for different wire widths.
4) Nets should be allowed to have floating terminals so that the router can select the best terminal locations.
5) In case the cell is unroutable, the router should be allowed to have some pins connected externally. The number of unconnected pins should be minimized.
6) Recommended design rules to improve yield should be allowed. One of the targets should be the maximization of the compliance of recommended design rules.
7) Wire length should be a parameter for optimization.

C. Goals and Contributions

The primary goal of this paper is to propose an algorithmic approach for the generic problem of cell routing with the properties described in Section I-B.

Another important goal is efficiency. The proposed router should be able to synthesize a complete standard cell library, including complex sequential cells (e.g., scan flip-flops). In this way, various layout templates using different number of tracks and transistor widths can be explored and area/performance/power metrics can be quickly evaluated.

Similarly to [7] and [8], the approach presented in this paper is based on a generic Boolean formulation of the problem. It uses the LDP paradigm to make the router technology independent and the design rules parameterizable. The main contributions of this new approach are as follows:

1) a new encoding scheme for SAT-based formulas that makes the problem tractable for large cells;
2) new graph-theoretical results to support floating terminals, which are essential to route most of the nets in standard cells;
3) efficient windowing heuristics for incremental routing of nets that allows to complete the routing for large cells;
4) an algorithmic-independent formalism to specify gridded design rules and multiple-patterning constraints;
5) a strategy to allow externally-connected pins in case the cell is unroutable;
6) a heuristic strategy for quality improvement (wirelength and recommended design rules).

The combination of the previous contributions has enabled the synthesis of a library with 127 standard cells in about one hour and a half of CPU time. The synthesis results of the complete library can be found in .

The paper is organized as follows. Section II reviews existing approaches for cell routing. Section III presents a graph model for the routing problem and a Boolean representation of the space of solutions. Section IV introduces the Boolean constraints that model the routing problem, whereas Section V introduces a Boolean formulation for the parameterizable design rules. Section VI describes the optimization phase of the routing algorithm. The experiments to synthesize a cell library are reported in Section VII.

II. Cell Routing

Detailed routing is commonly considered as the problem following global routing in which the routes of the nets must be assigned to metal segments and vias. Conceptually, global and detailed routing are similar problems and use related algorithmic approaches.

The most classical algorithms for detailed routing were designed by assuming that design rules would be implicitly honored by defining a sufficiently large technological pitch for the underlying routing grid. For standard cells, in which area is limited by the cell height, compaction was often integrated to optimize area [3].

Nowadays, the complexity of the design rules is growing exponentially with each technology node. Design rules are more context-sensitive and the window of influence of each checked pattern is expanding. For this reason, manual design of standard cells requires a titanic effort and design automation becomes essential.

Different options can be envisioned to automatically generate DRC-compliant circuits:

1) defining an overly conservative technological pitch for the underlying grid in a way that the spacing rules can be ignored during detailed routing;
2) iterating detailed routing and DRC by ripping-up and rerouting conflicting nets;
3) constructing DRC-compliant circuits by integrating manufacturing constraints in the detailed router.

Option 1) may imply an area overhead with unacceptable impact on manufacturing cost and yield. Option 2) may be acceptable for noncongested routing regions [11] but may not find an existing feasible solution for cells with congested regions that require sinuous routes for a complete routability.

As the complexity of design rules grows, option 3) seems to be the only one that can guarantee the synthesis of area-efficient solutions for high-density DRC-compliant

circuits. That is the main reason why, in the last few years, we have seen an increasing effort in integrating manufacturing constraints in cell synthesis tools [7], [11]–[15]. However, this integration implies a significant computational cost.

The main goal of this paper is to propose a detailed routing approach that is computationally affordable when integrating manufacturing constraints.

A. Concurrent Detailed Routing

There are two main strategies for routing according to the approach used to create the routes for each net: sequential (one net at a time) and concurrent (all nets simultaneously) [16].

For similar reasons as the ones mentioned above, sequential routing may not guarantee a feasible solution even if it exists, specially for high-density cells. For this reason, the most recent approaches resort to schemes for concurrent routing.

There are two families of algorithms that can be considered for concurrent routing depending on the objects used to take the routing decisions: tree-based and segment-based.

Typically, tree-based algorithms work in two steps. First, a set of candidate routing trees is generated for each net using algorithms for the generation of multiple rectilinear Steiner minimum trees (e.g., as in [17]). Next, a multicommodity flow problem is formulated and solved as a 0-1 integer linear programming problem.

Segment-based algorithms work at the level of individual metal segments. Given the fine granularity of the decisions, the problem becomes computationally more complex than tree-based algorithms. However, the formulation of the problem is much more amenable for the incorporation of manufacturing constraints. Usually, SAT-based formulations are used in which every wire segment is represented as a Boolean variable [7], [8].

Tree-based approaches using SAT formulations have been proposed in [15] and [18]. In [15], a tree-based approach at a fine level of granularity is proposed. For every net, a maze router generates a set of routes for all pairs of terminals. The maze router is process-dependent, meaning that it must be customized for a specific technology. Some of the design rules are checked on-the-fly when the routes are generated. A Boolean formulation guarantees the selection of a set of routes that creates a tree connecting all terminals. Design rules are modeled as conflicts between sets of routes.

The approach in [18] combines a segment-based with a tree-based formulation for routing double-gate, transistor-array-based layouts. To avoid the explosion of potential routing paths, a greedy phase is used to preroute some nets. The number of turns for each path is also limited to prevent the usage of sinuous paths.

To some extent, the approaches in [15] and [18] are similar in the sense that the branching decision during SAT are taken based on the selection of paths (and not wires). The formulation in [15] seems to be simpler and more effective.

In general, tree-based approaches are convenient when the set of possible routes is not large and the design rule conflicts can be modeled between pairs of nets. In case of an explosion of routes, some pruning can be applied at the expense of declaring a problem unroutable even if a solution exists. The models can become much more complex when dealing, for example, with floating terminals since there can be an explosion of routes to connect different regions of the layout.

The routing models can also become complex when design rules involving several nets are required. For example, multiple-patterning rules require the analysis of multiple nets. This analysis can be complex if the set of valid routes for each net is large.

The closest approach to the work presented in this paper was proposed in [7]. It is a segment-based approach of the routing problem inspired by the satisfiability formulation presented in [8]. The design rules are embedded in the model as Boolean constraints. A solution of the model represents a valid set of routes and an unsatisfiable model corresponds to an unroutable cell. However, no chance is given to provide a partial routing with externally connectible pins.

The approach presented in [7] showed a high computational complexity and only small gates could be routed. A comparative analysis with the approach presented in this paper will be discussed in Section VII.

B. Restrictive Design Rules and Pattern Constructs

At every technology node, the number and complexity of design rules are growing exponentially. For a tractable automation, they are usually discretized and adapted to gridded layouts, thus generating a set of conservative restrictive design rules (RDRs) [19], [20] extracted from the gridless design rules associated to the technology. Obtaining a set of RDRs is mostly a manual task based on the knowledge of manufacturability constraints and the structure of the underlying coarse grid of the layout. Automating the generation of RDRs from gridless design rules is a desirable feature of the synthesis flows and a topic for research.

In this paper, we consider the use of RDRs that can be sufficiently generic to cover specific manufacturing aspects of the technology. For example, the use of different metal widths or multiple-patterning lithography (MPL) can be easily supported by encoding different attributes in the Boolean formulation of the problem.

The specification of RDRs is highly simplified when MPL constraints are provided. Usually, MPL constraints can have simple specifications based on the definition of minimum-distance rules [21] and enable to simplify the RDRs by ignoring those patterns that do not honor the MPL constraints, i.e., MPL violations can be considered as don't care conditions for the rest of RDRs.

The specification of design rules with Boolean formulas will be discussed in Section V.

III. Routing Problem

Graphs are the natural formal model to represent a gridded routing problem. Fig. 2(a) shows a particular instance of the routing problem with two nets represented with rectangular and oval boxes, respectively. Every net has a set of terminals that must be connected. Each terminal is represented by a set of vertices. Terminals with more than one vertex represent floating terminals in which every vertex is a possible

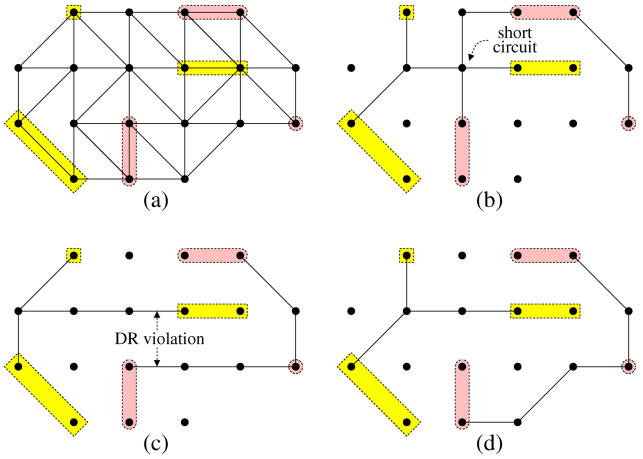


Fig. 2. (a) Graph formulation of a routing problem with two nets. (b) Illegal solution with a short circuit. (c) Illegal solution with a design rule violation. (d) Legal solution.

connection point. Edges represent wire segments that can be used to connect pairs of vertices. The routing problem can be formulated as follows:

> *Find a set of edges that define routes connecting the terminals of each net. The routes must be disjoint (cannot have common vertices) and satisfy a set of design rules.*

Let us assume that we have only one design rule that prohibits the use of parallel edges that do not have any other edge in between. Fig. 2(b) depicts an illegal solution since the two routes have a common vertex. The routes in Fig. 2(c) are disjoint; however, they violate the design rule. Finally, Fig. 2(d) depicts two routes that conform a valid solution for the problem.

It is important to realize that the number of possible solutions for a gridded routing problem is finite, since the underlying graph is also finite. Finding a legal solution for a routing problem can be reduced to a Boolean satisfiability problem in which a variable is associated to every edge, representing the presence or absence of a wire.

The main goal of the approach presented in this paper is to find a legal routing that maximizes the quality of the solution using a satisfiability (SAT) formulation. SAT is NP-complete, but efficient solvers can still provide solutions for large problems in an affordable time.

Different aspects can be considered when evaluating the quality of a solution: wirelength, number of unconnected terminals, compliance of recommended design rules, etc. Unfortunately, the optimization extensions of SAT (e.g., MAXSAT) are NP-hard and soon become intractable for many problems in which the decisional version is still tractable.

For the previous reason, we propose to solve the routing problem in two steps:

1) finding a legal solution that honors the design rules;
2) improving the solution by iteratively rerouting nets and using quality terms in the cost function.

This section and Sections IV and V cover the first step. Section VI covers the second step.

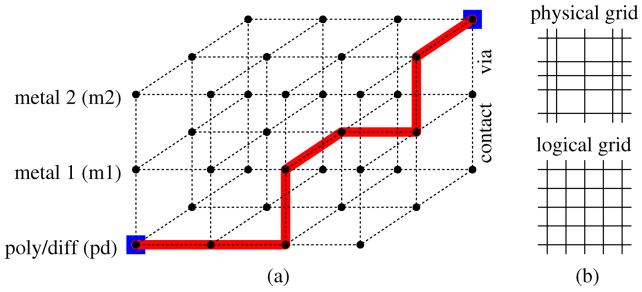
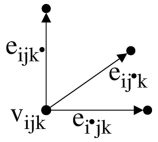


Fig. 3. (a) Grid model for routing. (b) Physical and logical grid.

TABLE I
NOMENCLATURE USED FOR OBJECTS IN ROUTING MODEL

Symbol	Description
$u, v, \ldots$	Vertices of the grid
$e, e', \ldots$	Edges of the grid
$n, n', \ldots$	Nets
$s, s', \ldots$	Subnets



The algorithms in this paper have been used for the synthesis of library cells. However, the approach can be used for any general routing problem.

A. Representation of Routing Region and Nomenclature

The interconnect model is similar to the one presented in [8]. For the sake of simplicity in the nomenclature and formulation of the problem, we customize the model for 3-D grids. However, the reader will realize that the model can be easily extended to any graph model, as it was shown in the example of Fig. 2.

The routing region is represented by a 3-D undirected grid graph $G(V, E)$, as depicted in Fig. 3(a), where the vertices (grid points) have associated integer coordinates in $\{1, \ldots, W\} \times \{1, \ldots, L\} \times \{1, \ldots, H\}$, where W, L and H are the width, length, and height of the grid, respectively. The edges of the graph connect grid points. Even though the logical grid assumes unit-length edges, the represented physical grid may have rows and columns separated by different distances, as depicted in Fig. 3(b). This heterogeneity will be taken into account when instantiating the design rules at every grid point (Section V).

Every vertex v of the grid is denoted by its coordinates $v = (x, y, z)$. In our context, the coordinate z represents the layer of the layout, e.g., $z \in \{\texttt{pd}, \texttt{m1}, \texttt{m2}\}$, as depicted in Fig. 3(a).

Table I summarizes the nomenclature used for the grid and routing objects. The column of symbols describes the range of letters used to denote every type of object. The subscripts indicate the coordinates of the vertices (v_{ijk}) and the origin points of the edges.

Edges will be denoted by their endpoints, e.g., $e(u, v)$, or by the coordinates of one endpoint and the direction of the edge in the grid, as shown in Table I. The nomenclature $e_{i\bullet jk}$, $e_{ij\bullet k}$, and $e_{ijk\bullet}$ is used to denote edges starting at vertex v_{ijk}. The superscript $\bullet$ indicates the dimension occupied by the edge, as

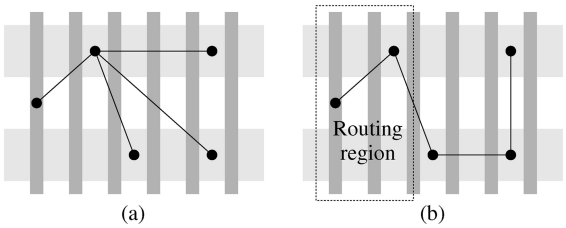


Fig. 4. Different connectivity models for subnets. (a) Star. (b) EMST.

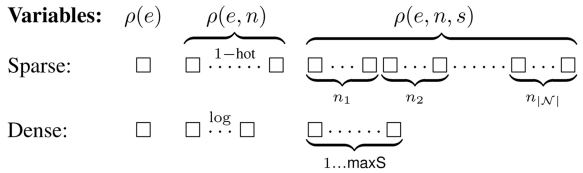


Fig. 5. Encoding schemes for the formulation of the routing problem (every box represents a Boolean variable).

shown in Table I (right). For example, vias will be represented by vertical edges that start at the m1 level ($e_{ij\mathsf{m1}\bullet}$).

An edge $e = \{u, v\}$ is said to be adjacent to vertices u and v. Two edges are said to be adjacent if they share a common vertex. The set of edges adjacent to a vertex v or edge e will be denoted by adj(v) or adj(e), respectively. To simplify the nomenclature, we use quantifiers ($\forall v$, $\forall e$, ...) without specifying the domain, that will be implicit with the name of the variable ($\forall v \in V$, $\forall e \in E$, ...).

B. Nets and Subnets

A net $n \subset V$ is a set of grid points called terminals that must be connected. A routing problem $\mathcal{N}$ is defined by a set of nets $\mathcal{N}$. The routing problem consists of finding routes along the edges of the grid that connect all terminals of each net. The routes must be compliant with a set of design rules.

A subnet is a pair of terminals of the same net. Finding a routing for an n-terminal net can be reduced to the problem of finding a routing for $n - 1$ subnets that connect all terminals [8]. In [8] and [22], a star model was proposed to connect all terminals, as shown in Fig. 4(a), in which one of the terminals was shared by all subnets.

We propose an alternative model based on finding an Euclidean minimum spanning tree (EMST). This model guarantees the connectivity of the net and minimizes the length of the edges [see Fig. 4(b)]. The effects of using this model to reduce the complexity of the problem are discussed in Section IV-C.

C. Boolean Variables and SAT Formulation of Problem

The problem encoding has a direct impact on the size of the model and the computational effort to solve it. Finding a good set of variables and clauses can have a significant impact on the behavior of a SAT solver [23]. On one hand, minimizing the number of variables may result in larger formulas and worse performance. However, smaller formulas do not always guarantee better performance. It is usually convenient to find encodings in which unit propagation can be efficiently exploited by the DPLL algorithm [24].

In this paper, we have evaluated different encodings to represent a routing solution but we finally compare two of them: the one proposed in [8] and [22] and the one proposed in this paper that we call dense and sparse encodings, respectively. The encodings represent the wires (edges) in the grid and their associated nets and subnets. An important parameter of the model is the maximum number of subnets of a net

$$\mathsf{maxS} = \max_{n \in \mathcal{N}} |n| - 1.$$

The symbol ρ is used to denote Boolean variables, abusing from symbol overloading to simplify the nomenclature.

The dense encoding uses the smallest number of variables, whereas the sparse encoding uses the largest. For every edge e of the grid, three sets of variables are used (see Fig. 5).

1) $\rho(e)$: a variable that represents when e is occupied by a wire. This variable was not used in [8] but becomes essential for the specification of design rules.
2) $\rho(e, n)$: a set of variables encoding the associated net in case e is occupied by a wire. A one-hot encoding is used for the sparse encoding (one variable for every signal), whereas a log-encoding is used for the dense encoding ($\lceil \log_2 |\mathcal{N}| \rceil$ variables).
3) $\rho(e, n, s)$: a set of variables to encode the subnets associated to every wire. The dense encoding use maxS variables for every edge. The sparse encoding uses one variable for each possible pair (net, subnet) of the routing problem.

We will say that e is a wired edge when $\rho(e)$ is asserted. A wire segment is a sequence of adjacent wired edges.

Every edge can hold more than one subnet of the same net. Therefore, the $\rho(e, n, s)$ variables are not one-hot encoded. For the sparse encoding, every net has a set of subnet variables. Since every edge can only hold one net, only the variables of one of the sets can be asserted. Even though this seems to be an inefficient encoding scheme, it turns out to be more computationally efficient in practice.

The $\rho(e, n, s)$ variables will be used for the routability formula, whereas the $\rho(e, n)$ and $\rho(e)$ variables will be used for the specification of the design rules.

Hence, for the sake of simplicity, all the constraints of the Boolean model will be described for the sparse encoding. The formulation for the dense encoding is conceptually similar. The reader can resort to [8] and [22] for the details of the dense encoding.

The routing problem is represented by a Boolean formula $\mathcal{F}$ with three types of constraints

$$\mathcal{F} \equiv \mathcal{C} \ \wedge \ \mathcal{R} \ \wedge \ \mathcal{DR} \tag{1}$$

where $\mathcal{C}$ represents the set of consistency constraints (Section IV-A), $\mathcal{R}$ represents the routability constraints (Section IV-B), and $\mathcal{DR}$ represents the design-rule constraints (Section V).

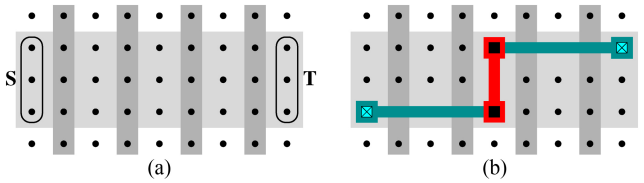


Fig. 6. (a) Specification of floating terminals. (b) Valid solution.

IV. Consistency and Routability Constraints

A. Consistency Constraints

We next present the formulas that define relationships between the variables $\rho(e)$, $\rho(e, n)$ and $\rho(e, n, s)$. Two constraints are required.

1) If an edge is assigned to a net, then the edge must be occupied by a wire

$$\forall e : \bigvee_{n \in \mathcal{N}} \rho(e, n) \Longrightarrow \rho(e). \tag{C1}$$

2) If an edge is associated to some subnet of a net, it must be also associated to the net:

$$\forall e, \forall n : \bigvee_{s \in subnets(n)} \rho(e, n, s) \Longrightarrow \rho(e, n). \tag{C2}$$

These constraints are formulated as implications ($\Rightarrow$) and not equivalences ($\Leftrightarrow$) since they can be represented with simpler CNF formulas. The equivalences are not necessary. In some cases, a wire could appear in some edge without being assigned to any net. These cases are not harmful since they do not restrict the satisfiability of the problem. On the other hand, the spurious wires will be cleaned during the optimization phase.

More interestingly, there can be wires assigned to a net but not assigned to any subnet. These cases are necessary. For example, some design rules require the extension of the wire segments to guarantee a minimum segment length. These extensions may violate the routability rules of the subnets (discussed in Section IV-B). Therefore, the unidirectional implication in (C2) is not only convenient, but necessary.

B. Routability Constraints With Floating Terminals

This section presents a new routability model inspired by the model presented in [8]. In this paper, the model is extended to support floating terminals, which is an essential feature to exploit the routing flexibility within the cells.

The interconnection in library cells typically requires routes between regions of grid points, e.g., a region of diffusion connected to a region of polysilicon, or a region of diffusion connected to a region of metal 2 to implement an I/O pin. In general, routing has to deal with floating terminals located in regions of grid points. An example is shown in Fig. 6 in which the routing of a subnet must guarantee the connectivity between two terminals in the regions S and T.

Our model highly simplifies the routability constraints presented in previous approaches. We resort to the theory by Euler from his famous paper on the *Seven Bridges of Königsberg* [25]. From Euler's theory we know:

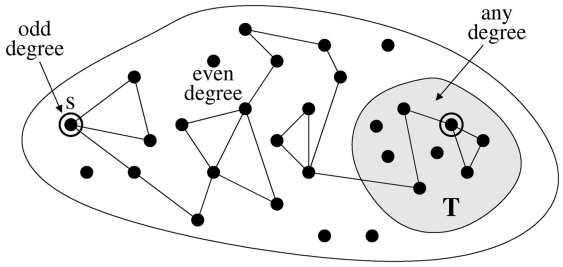


Fig. 7. Illustration of Theorem 2.

Theorem 1 (Handshake Theorem): The sum of degrees of all vertices of a graph is an even number.

Corollary 1: The number of vertices of odd degree is even.

We now extend Euler's theorem to support floating terminals.

Theorem 2 (Path with one floating terminal, see Fig. 7): Let $G = (V, E)$ be a graph such that $s \in V$ is a vertex with odd degree, $T \subset V$ is a set of vertices with arbitrary degree and $V' = V \setminus (T \cup \{s\})$ are the remaining vertices, all of them with even degree. Then, G has a path that joins s with some vertex $t \in T$.

Proof: Let us call G_s the connected component of G that includes vertex s. The degree of the vertices in G_s is the same as the degree of the same vertices in G, since only vertices disconnected from the connected component are removed. By using Corollary 1 we conclude that G_s must have at least one vertex $t \in T$ with odd degree, otherwise G_s would have one and only one vertex with odd degree. Since G_s is connected, there is a path from s to t. ■

The previous theorem can be extended for paths with two floating terminals.

Definition 1 (External edges): Given a subset of vertices $S \subset V$, $e = \{u, v\}$ is said to be an external edge of S if $u \in S$ and $v \notin S$. We call $\mathsf{Ext}(S)$ the set of external edges of S.

Theorem 3 (Paths with two floating terminals): Let $G = (V, E)$ be a graph. Let $S \subset V$ be a set of vertices such that $|\mathsf{Ext}(S)|$ is odd. Let $T \subset V$ be a set of vertices with arbitrary degree and $V' = V \setminus (T \cup S)$ are the remaining vertices, all of them with even degree. Then, G has a path that joins a vertex $s \in S$ with a vertex $t \in T$.

Proof: The handshake theorem indicates that the sum of degrees of the vertices in S is odd. Let us collapse all nodes of S into one node $\hat{s}$, preserving all edges even in the case they are self-loops or replicated. Now $\hat{s}$ has a degree equal to the sum of degrees of the original vertices in S. Therefore, $\hat{s}$ has odd degree. We are now in the same conditions of Theorem 2, thus guaranteeing there is a path from $\hat{s}$ to a vertex $t \in T$. This also guarantees a path from S to T in the original graph. ■

Corollary 2: Let $G = (V, E)$ be a graph and $S \subset V$ be a set of vertices with $|\mathsf{Ext}(S)| = 1$. Let $T \subset V$ be a set of vertices with arbitrary degree and $V' = V \setminus (S \cup T)$ are the remaining vertices, all of them with degree zero or two. Then, G has a path that joins a vertex $s \in S$ with a vertex $t \in T$.

Proof: It is just a particular case of Theorem 3. ■

The previous results allow to formulate the routability constraints of a subnet as a conjunction of local properties on

the vertices (odd/even degree). It also allows to have floating vertices on every subnet.

The properties on every vertex can be defined in terms of Theorem 3 or Corollary 2. The latter formulation leads to solutions with simple paths (nonrepeated vertices) and smaller Boolean formulas. For this reason, the formulation based on Corollary 2 is preferred.

Let us now consider the routability constraints for a subnet s of a net n. The subnet has to connect two regions, S and T. We define a proposition to enforce one external edge on a set S of grid points

$$\mathsf{Terminal}(S, n, s) \equiv \sum_{e \in \mathsf{Ext}(S)} \rho(e, n, s) = 1$$

where the summation represents the number of asserted variables.

We introduce some expression to denote the number of wired edges connected to a grid point (degree of the grid point)

$$\mathsf{Nadj}(v, s, n) \equiv \sum_{e \in \mathsf{adj}(v)} \rho(e, n, s).$$

The next proposition enforces a grid point with degree zero or two

$$\mathsf{Degree0or2}(v, n, s) \equiv \mathsf{Nadj}(v, n, s) = 0 \ \vee \ \mathsf{Nadj}(v, n, s) = 2.$$

The previous equalities can be easily transformed into Boolean clauses using different methods (e.g., BDDs, adders, sorters [26]).

The constraints for the routability of the subnet are as follows:

$$\mathsf{Terminal}(S, n, s) \tag{R1}$$

$$\forall v \notin (S \cup T) : \mathsf{Degree0or2}(v, n, s). \tag{R2}$$

Note that no constraints are imposed on the grid points of T. Given the symmetry of the formulation, we could also choose to impose the constraints on T and leave S unconstrained. We choose to add the constraints for the smallest set, resulting in smaller Boolean formulas.

The previous constraints allow each edge to be assigned to more than one subnet, which is acceptable if the subnets belong to the same net. Additional constraints are required to ensure that every edge is assigned to one net at most and two adjacent wires are assigned to the same net (otherwise a short-circuit would be produced)

$$\forall e, \forall n, \forall n' \neq n : \rho(e, n) \Longrightarrow \neg\rho(e, n') \tag{R3}$$

$$\forall n, \forall e, \forall e' \in \mathsf{adj}(e) : \rho(e, n) \wedge \rho(e') \Longrightarrow \rho(e', n). \tag{R4}$$

For those nets requiring an input/output pin, a new subnet is created that connects one of the poly/diffusion regions of the net with a floating terminal in the uppermost layer (usually metal 2).

C. Subnet Windowing

Empirically, it is easy to observe that the route of a two-terminal subnet rarely spans beyond the bounding box determined by the two terminals. For this reason, restricting the routability of a subnet to a region around the bounding box is a strategy that contributes to reduce the complexity of the problem.

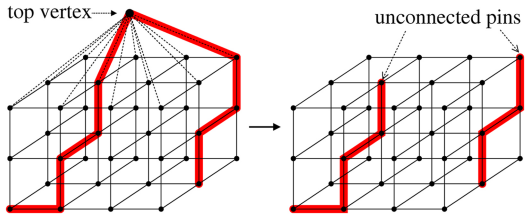


Fig. 8. Using a top vertex to implement unconnected pins.

Fortunately, the sparse encoding helps to define restricted routing regions for each subnet by simply dropping those variables that do not belong to the routing region. Formally, this is equivalent to enforcing the variables outside the region to be falsified, i.e.

$$\forall e, \forall n, \forall s, \ \text{such that } e \notin \mathsf{Region}(n, s) : \ \neg\rho(e, n, s).$$

The elimination of the subnet variables is not possible when using the dense encoding since these variables are associated to all nets simultaneously.

The routing problem uses a parameter $\mathcal{W}$ that determines the expansion of the routing region around the bounding box of each subnet. Formally, the region of the grid (window) to route a subnet is composed of all the edges inside the bounding box and those that are not farther than $\mathcal{W}$ units from the bounding box.

In practice, the window needs to be expanded few grid points beyond the bounding box of each subnet. When using windowing, there is a probability that no solution is found even one exists. However, for large cells, windowing often transforms the problem from intractable to tractable and, therefore, losing some solution is the lesser of two evils.

Windowing is the reason why an EMST is calculated for each net to determine the set of two-terminal subnets (see Section III-A and Fig. 4). Using EMST guarantees that the sum of the areas of the bounding boxes is minimized.

D. Unconnected Pins

Some unroutable cells may become routable if some pins are left unconnected and their connection is delegated to the upper metal layers. This option is not desirable, but necessary if no routing solution is found. We next describe how the problem of minimizing the number of unconnected pins can be solved algorithmically.

Fig. 8 depicts the grid model used for the case in which one pair of unconnected terminals is allowed. A new point is added on top of the grid graph connected to all the points of the upper layer, thus forming a pyramidal ceiling with a top vertex accepting the routing of one of the subnets. The two edges of the pyramid used for routing will determine the unconnected pins of the cell.

This strategy can be extended to any number of unconnected pins by allowing more subnets to be routed through the pyramidal ceiling and adding a constraint on the number of routed subnets.

Experimentally, we have observed that most of the cells can be completed without unconnected pins. Only few congested cells require some unconnected pins. In case the original problem is unsatisfiable, one subnet is allowed to be routed through the pyramidal ceiling. If still unsatisfiable, two subnets are allowed, etc.

V. Design Rules

This section explains how design rules can be specified in a grid-based router using a Boolean formulation. We first present few examples and discuss the extension of this formulation to handle attributes on the wires. The use of attributes will allow, for example, to define design rules for multiple patterning lithographies.

A technology-independent specification language for design rules in gridded layouts has been created. It is based on a simple subset of C++ syntax. In this way, the design rule file can be compiled into a dynamic library that can be loaded at execution time.

The design rule language can specify attributes for the wire segments and Boolean predicates that model the allowed patterns in the layout. The same language is used to specify mandatory (hard) and recommended (soft) design rules.

Design rules are specified with two parameters, i and j, that refer to a reference location in the grid graph. Every design rule is expanded across the grid graph by instantiating all possible values for i and j in the grid. The design rule language also allows the specification of numerical constraints that restrict the application of the rules to specific locations, e.g., only for locations in which j is even ($j \mod 2 = 0$), at the left boundary of the cell ($i = 0$), etc.

Design rules can be specified as Boolean formulas only using the $\rho(e)$ and $\rho(e, n)$ variables. Despite that we shall not detail this language here, the curious reader may see an example of a design rule file in the design rules link of the web site. Let us discuss a few examples below.

A. Design Rule Examples

Example 1: *Metal 1 can only be laid out in the horizontal direction*

$$\forall e_{ij\bullet\mathsf{m1}} : \neg\rho(e_{ij\bullet\mathsf{m1}})$$

where $e_{ij\bullet\mathsf{m1}}$ denotes the edge in the vertical direction with origin in (i, j) using layer $\mathsf{m1}$ (see Table I).

Example 2: *Any horizontal wire segment in metal 1 must have length not shorter than two units*

$$\forall e_{i\bullet j\mathsf{m1}} : \rho(e_{i\bullet j\mathsf{m1}}) \implies \big(\rho(e_{i+1\bullet j\mathsf{m1}}) \vee \rho(e_{i-1\bullet j\mathsf{m1}})\big).$$

This constraint enforces any horizontal wired edge in $\mathsf{m1}$ to have an adjacent segment, thus ensuring a minimum length of two edges.

Example 3: *No adjacent vias can be connected to different nets*. Given two edges, e_1 and e_2, we can define a proposition that is asserted when they are assigned to the same net

$$\mathsf{SameNet}(e_1, e_2) \equiv \bigvee_n \big(\rho(e_1, n) \wedge \rho(e_2, n)\big).$$

The constraint to prevent adjacent vias is as follows:

$$\begin{aligned}\forall e_{ij\mathsf{m1}\bullet} : \rho(e_{ij\mathsf{m1}\bullet}) \implies \big(&\\ &\big(\neg\rho(e_{i-1\,j\mathsf{m1}\bullet}) \vee \mathsf{SameNet}(e_{ij\mathsf{m1}\bullet}, e_{i-1\,j\mathsf{m1}\bullet})\big) \wedge\\ &\big(\neg\rho(e_{i+1\,j\mathsf{m1}\bullet}) \vee \mathsf{SameNet}(e_{ij\mathsf{m1}\bullet}, e_{i+1\,j\mathsf{m1}\bullet})\big) \wedge\\ &\big(\neg\rho(e_{ij-1\mathsf{m1}\bullet}) \vee \mathsf{SameNet}(e_{ij\mathsf{m1}\bullet}, e_{ij-1\mathsf{m1}\bullet})\big) \wedge\\ &\big(\neg\rho(e_{ij+1\mathsf{m1}\bullet}) \vee \mathsf{SameNet}(e_{ij\mathsf{m1}\bullet}, e_{ij+1\mathsf{m1}\bullet})\big)\big).\end{aligned}$$

This constraint indicates that any via in point (i, j) implies no vias in the four adjacent grid points, unless the vias are connected to the same signal. A more restricted version could be considered by involving the eight edges surrounding the via in (i, j).

B. Assigning Attributes

For generating a gridded layout, it may not be sufficient to only define the presence or absence of wires in the edges of the underlying graph. Some physical aspects of the wires may also be required for a complete description. For example, wires might have two different widths (thin and thick) with different associated design rules or could be assigned to different masks to comply with multiple patterning lithography rules.

Any discrete set of attributes can also be binary-encoded and incorporated in the Boolean formulation of the problem. We propose to use a one-hot encoding for sets of attributes associated to wires. We will denote by $\rho(e, x)$ the variable that represents the presence of attribute x in edge e.

For example, if wires can have two different widths and can be assigned to k different masks, the following variables can be defined for each edge e:

$$\rho(e, \mathtt{thin}), \quad \rho(e, \mathtt{mask}_1), \ \rho(e, \mathtt{mask}_2), \ldots, \rho(e, \mathtt{mask}_k).$$

The assertion of $\rho(e, \mathtt{thin})$ will indicate that the wire is thin, otherwise the wire is thick. Note that the variables representing the masks are not necessarily mutually exclusive, thus allowing overlapping masks. Mutual exclusion between attributes can always be enforced by Boolean constraints.

C. Multiple Patterning Lithography

We next illustrate how attributes can be used to formulate multiple patterning lithography (MPL) rules and enforce MPL-compliant layouts.

Not every routing solution may be feasible when a limited number of masks can be used. Fig. 9 shows a possible routing for the horizontal metal layer that cannot be implemented with only one mask if the different wire segments must be separated by a distance greater than d. However, the problem is solvable with two masks.

The Boolean formulation of the routing problem can be extended to generate MPL-compliant layouts using the $\rho(e, \mathtt{mask}_i)$ variables.

Let us consider the MPL assignment for layer $\mathtt{m1}$ (metal 1). with the set of masks $\mathcal{M} = \{\mathtt{mask}_i\}$.

For a consistent encoding, these mask variables need to be related to the routing variables

$$\forall e_{ij\mathsf{m1}} : \rho(e_{ij\mathsf{m1}}) \implies \bigvee_{\mathtt{mask}_i \in \mathcal{M}} \rho(e_{ij\mathsf{m1}}, \mathtt{mask}_i)$$

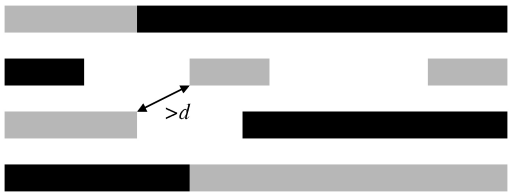


Fig. 9. Example of double-patterning lithography.

indicating that every wired edge must be assigned to one mask at least. Note that this formulation does not prevent a wired edge to be assigned to more than one mask, thus resulting in overlapping segments.

Let us illustrate with an example, how constraints for MPL could be specified for m1, assuming that m1 can only be used for horizontal routing and any pair of wire segments must be separated by a distance greater than d.

We first introduce a new predicate to represent the fact that two edges are assigned to the same mask

$$\mathsf{SameMask}(e_1, e_2) \equiv \bigvee_{m_i \in \mathcal{M}} \rho(e_1, \mathtt{mask}_i) \ \wedge \ \rho(e_2, \mathtt{mask}_i).$$

We specify the rule by considering two cases, depending on whether the wire segments are in the same or in different tracks. When belonging to different tracks, wire segments not sufficiently spaced must be in different masks

$$\forall e_{i_1^\bullet j_1 \mathsf{m1}}, e_{i_2^\bullet j_2 \mathsf{m1}}, \quad \text{such that} \quad j_1 \neq j_2, \quad \mathrm{Distance}(e_{i_1^\bullet j_1 \mathsf{m1}}, e_{i_2^\bullet j_2 \mathsf{m1}}) \leq d : \neg \mathsf{SameMask}(e_{i_1^\bullet j_1 \mathsf{m1}}, e_{i_2^\bullet j_2 \mathsf{m1}}).$$

When belonging to the same track, disconnected wire segments that are too close must be in different masks. Wire segments are disconnected when there is some empty slot between both

$$\forall e_{i_1^\bullet j \mathsf{m1}}, e_{i_2^\bullet j \mathsf{m1}}, \quad \text{such that} \quad i_1 < i_2 - 1, \quad \mathrm{Distance}(e_{i_1^\bullet j \mathsf{m1}}, e_{i_2^\bullet j \mathsf{m1}}) \leq d : \bigvee_{i_1 < i < i_2} \neg \rho(e_{i^\bullet j \mathsf{m1}}) \implies \neg \mathsf{SameMask}(e_{i_1^\bullet j \mathsf{m1}}, e_{i_2^\bullet j \mathsf{m1}}).$$

The important fact of this approach is that by incorporating the MPL constraints into the Boolean formula, all the generated layouts are guaranteed to be MPL-compliant.

VI. Routing Optimization

The Boolean formula (1), if satisfiable, generates a valid routing for the problem, which is the main goal of the cell router. Among all valid routing solutions, the ones with the best quality are desired. There are several parameters that can be taken into account to improve the quality of a layout. For example, cells with smaller wirelength are preferred. For reliability reasons, which have a direct impact on yield, some layout patterns are recommended, such as the addition of redundant vias.

When the routing model is extended with a cost function, the model moves from the NP-complete (SAT) to the NP-hard domain (MAXSAT, ILP, etc), thus involving a substantial jump in computational complexity. This section proposes a heuristic method to make this problem tractable using 0-1 linear programming (0-1 LP).

Any SAT model can be reduced to a 0-1 LP model by transforming every clause $(x_1 \vee \cdots \vee x_n)$ in the CNF formula into a linear inequality[1]

$$\sum_{i=1}^{n} x_i \geq 1$$

with x_i being 0-1 variables. We next propose a model to improve the quality of the routing solution.

A. Optimization of Recommended Design Rules

Let us consider a set of recommended rules $\mathcal{RR} = \{r(i, j)\}$ applied to every point (i, j) of the grid. For every rule $r(i, j)$, we have an estimation of the cost for violating the rule (e.g., yield loss) that can be quantified as a real number α_r.

With an abuse of notation, let us consider $r(i, j)$ to be the Boolean formula representing the same recommended design rule. A new variable v_{ij}^r is created for each (i, j)-instance of $r(i, j)$ to denote the violation of the rule, i.e.

$$\forall i, j: \quad \overline{r(i, j)} \implies v_{ij}^r.$$

The 0-1 linear programming model can be formulated as

$$\begin{aligned} \min \quad & \sum_{i,j,r} \alpha_r v_{ij}^r \\ \text{s.t.} \quad & \mathcal{F} \ \wedge \ \mathcal{RR} \end{aligned} \tag{2}$$

where $\mathcal{F}$ is formula (1) and $\mathcal{RR}$ is the Boolean formula representing all instances of the recommended design rules.

The previous model allows the violation of recommended rules at the expense of increasing the value of the cost function. Therefore, total weighted cost of the violations will be minimized.

It is important to realize that every feasible solution of formula $\mathcal{F}$ (1) is a feasible (possibly nonoptimal) solution of the 0-1 LP model (2), since the recommended design rules only affect the cost of the model, but not the space of solutions.

B. Examples of Recommended Design Rules

Let us consider two simple examples of recommended design rules: via minimization and addition of redundant vias. Via minimization is tightly related, and is a simplified version, of the wirelength minimization problem. For simplicity in the formulation, we focus only on via minimization.

Let us recall that vias are represented by vertical edges from m1, i.e., $e_{ij\mathsf{m1}^\bullet}$. The recommended design rule for via minimization (VM) can be specified as

$$\mathrm{VM}(i, j) \equiv \neg \rho(e_{ij\mathsf{m1}^\bullet})$$

indicating that it is not recommended to use a via in point (i, j). The Boolean constraint to represent the violation of the rule is

$$\neg \mathrm{VM}(i, j) \implies v_{ij}^{\mathrm{VM}}$$

[1]Any negative literal $\overline{x_i}$ is expressed as $(1 - x_i)$.

that is rewritten, by substitution, as[2]

$$\rho(e_{ij\mathsf{m1}^\bullet}) \implies v_{ij}^{\mathrm{VM}}.$$

Addition of redundant vias is used for reliability reasons. The recommended rule for redundant vias (RV) can be specified as follows:

$$\mathrm{RV}(i,j) \equiv \rho(e_{ij\mathsf{m1}^\bullet}) \implies \left(\rho(e_{i-1j\mathsf{m1}^\bullet}) \vee \rho(e_{i+1j\mathsf{m1}^\bullet}) \vee \rho(e_{ij-1\mathsf{m1}^\bullet}) \vee \rho(e_{ij+1\mathsf{m1}^\bullet})\right)$$

indicating that the presence of a via in (i, j) should imply the presence of some via in the four neighboring grid points. The Boolean constraint to represent the violation of the rule is

$$\neg\mathrm{RV}(i,j) \implies v_{ij}^{\mathrm{RV}}.$$

C. Making Optimization Model Tractable

The 0-1 LP model (2) becomes intractable when dealing with large cells. This section proposes a heuristic to maintain the problem tractable based on an incremental and iterative approach for optimization.

The approach is based on the combination of integer linear programming (ILP) with metaheuristics to solve large combinatorial problems [27]. In particular, large neighborhood search (LNS) techniques [28] are proposed to reduce the complexity of the problem at the expense of sacrificing optimality.

In short, the combination of ILP and LNS consists of iteratively solving instances of the ILP model by using a variable fixing strategy. At each iteration, a subset of variables of the model are fixed to the value of the current best solution. The optimization model only considers the remaining variables. Subsequent optimizations change the set of fixed variables until some convergence criteria is met. As an example, this strategy has been successfully adapted for the 0-1 knapsack problem [29].

We adapt the ILP&LNS strategy to the routing problem by mimicking a rip-up and reroute approach for individual nets. The algorithm is described in Fig. 10.

Initially, the nonoptimized routing problem (1) is solved using a SAT solver. $\mathcal{S}$ represents the solution (sets of values associated to the variables) in case the formula is satisfiable. Next, the 0-1 LP model is constructed according to model (2).

The two nested loops implement the ILP&LNS strategy. The innermost loop iterates over each net in the circuit. The function simplify calculates a simplified 0-1 LP model by fixing all variables with the values in solution $\mathcal{S}$, except the ones involving net n_i. The ILP solver finds the optimal solution for net n_i, under the assumption that all the other nets have been fixed. These two steps mimic the ripping-up and rerouting of net n_i. At this point it is important to realize that:

1) the model $\mathcal{M}'$ is always satisfiable, since the current solution $\mathcal{S}$ is a valid solution for the model;
2) the solution delivered by the ILP solver satisfies all the hard design rules.

[2]This is a particular case in which the rule can be represented with only one literal that could be used directly in the cost function without the need to create the new variable v_{ij}^{VM}.

```
S = SAT(F);                          ▷ Initial unoptimized solution
M = 0-1 LP routing model;            ▷ Model (2)
repeat
  for all n_i ∈ N do
    M' = SIMPLIFY(M, S, n_i);        ▷ Rip-up n_i
    S = ILP_SOLVER(M');              ▷ Re-route n_i
until no improvement
```

Fig. 10. IPL and LNS algorithm for routing optimization.

The rip-up and reroute application over all nets continues until no significant improvement is observed (outermost loop). In practice, two iterations of the outermost loop are sufficient to obtain a solution close to a local minimum.

This strategy admits several variants. For example, more than one net can be ripped-up and rerouted simultaneously, depending on the size of the problem. The order of the nets is also another aspect that can be explored to find better local minima.

VII. Experiments

In this section, we present several experiments that show the applicability of the routing algorithm.

A. Implementation and Experimental Setup

A routing tool has been implemented according to the algorithm presented in the paper. It uses PicoSAT [30] to solve the SAT models and Gurobi [31] to solve the optimization models. The CNFs have been generated by obtaining product-of-sums from BDD representations of the constraints [32] using the CUDD package [33].

A variation with dense encoding has also been implemented according to the constraints proposed in [22]. In this case, we have also included constraints for the external pins of the cells using the theory for floating terminals proposed in the current paper.

The experiments have been performed in an Intel(R) Xeon(R) X5670 CPU at 2800MHz using a single core and restricting the available memory to 4GB. A complete account of the experimental results can be found at the website: http://sites.google.com/site/cellrouting.

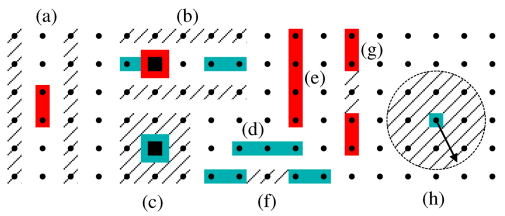
B. Benchmarks, Rules, and Synthesis Flow

The 127 cells of the Nangate 45nm open cell library [34] have been used to perform the experiments. Single- and double-height versions of the cells have been synthesized using two metal layers.

Specifically, polysilicon gates are equally spaced shapes placed vertically from top to bottom of the cell. All pMOS and nMOS transistors have a common width not being necessarily the same for both types of transistors. Narrow metal 1 is used for horizontal wires and wide metal 2 for vertical wires.

A set of commercial design rules has been adapted for gridded layouts using 1-D gridded design rules (only rectangular geometries [35]). The set of design rules is summarized in Table II and its actual formulation can be found at the design rules link of the web site.

TABLE II
DESIGN RULES USED FOR EXPERIMENTS

	
M1 can only be routed in the horizontal direction. M2 can only be routed in the vertical direction.	
Any vertical wired edge cannot have another neighboring wired edge in the six surrounding rows of the previous and following columns.	(a)
If two horizontal wired segments have two spaces in between and any of the endpoints is connected to a via, then no horizontal wires can be present in the five surrounding columns of the upper and lower rows of the edges.	(b)
A contact (or via) cannot have any other contact (or via) in the eight surrounding grid points.	(c)
Horizontal wired segments must have at least length 2.	(d)
Vertical wire segments must have at least length 3.	(e)
The space between two horizontal wire segments in the same row must be at least two grid units.	(f)
The space between two vertical wire segments in the same column must be at least two grid units.	(g)
DPL for M1: two different wire segments cannot share the same mask unless they are separated by two grid units.	(h)
IO-pins must be located in M2, in even columns.	

The flow to synthesize each cell is the following.

1) The SPICE transistor netlist is transformed by folding the large transistors and equalizing the width of all transistors in the same p or n strip. The transformation is done in such a way that the total width of each group of identical transistors is as similar as possible to the original width.
2) A transistor placement is computed using a placement tool that always guarantees an area-optimal layout. This tool has been designed based on the principles of [36].
3) A legal routing is computed using the SAT model described in Section III.
4) Wirelength is iteratively improved using the ILP&LNS heuristic described in Section VI.

Finally, the generated layouts are checked for DRC and LVS.

C. Comparison of Encoding Schemes

Our first experiment compares the dense and sparse encoding schemes evaluated in this paper. Table III reports some representative data of the Boolean formulation for the `FA_X1` (full adder) cell of the library. A window parameter of $\mathcal{W} = 6$ has been used in the sparse enconding.

As expected, the dense encoding uses fewer variables than the sparse encoding. Furthermore, it generates smaller formulas, both in the number of clauses and in the total number of literals. Despite of that, the SAT solver can solve the routing problem much faster with the sparse encoding than with the dense encoding.

TABLE III
ENCODING DIFFERENCES FOR FA_X1 AS SINGLE HEIGHT CELL

Encoding	Vars	Clauses	Size	Lits/cl	≤3 lits	CPU
Dense [22]	21k	103k	485k	4.7	40.0%	2959.0
Sparse	39k	359k	858k	2.4	98.5%	27.9

(*Size in literals, CPU time in seconds*)

TABLE IV
RESULTS FOR SAT SOLVING SINGLE HEIGHT CELLS

		Dense [22]		Sparse ($\mathcal{W} = 6$)	
Area	Cell	Size	CPU	Size	CPU
5	OAI221_X1	96k	0.1	158k	0.6
9	HA_X1	239k	29.4	295k	1.5
15	FA_X1	485k	2959.0	857k	27.9
21	DFFS_X1	903k	205.4	1429k	31.4
25	SDFF_X1	1380k	1424.0	2755k	84.9
33	SDFFRS_X2	2679k	>40 hours	4302k	142.1

(*Size in literals, CPU time in seconds*)

This is explained by two mutually related facts.

1) It is obvious that the windowing heuristic greatly reduces the search space. This heuristic cannot be applied with the dense encoding.
2) The structure of the clauses for each encoding is quite different: The dense encoding generates formulas with an average of 4.7 literals per clause, whereas this average is 2.4 for the sparse encoding. Likewise, 40% of the clauses of the dense encoding have two or three literals, whereas this percentage is 98.5% for the sparse encoding. Indeed, many constraints of the sparse encoding have the form $a \Rightarrow b$ or $a \wedge b \Rightarrow c$ and can be encoded with 2- and 3-literal clauses due to the one-hot encoding of the nets. The dense encoding requires more intricate formulas to deal with the log-encoding of the signals. Small clauses offer a great potential for unit propagation and simplification at every choice of the DPLL algorithm [37].

Similar conclusions can be drawn for all the other cells. Table IV reports results for some representative cells in the library, sorted in ascending order of area (number of polysilicon columns). It is clear that the dense encoding does not scale for medium and large cells. For instance, the `SDFFRS_X2` cell could not be routed in 40 hours with the dense encoding.

D. Wirelength Reduction

In order to show how the ILP&LNS heuristic works, Fig. 11 displays, for the `FA_X1`, the evolution of the total wirelength after cleaning each signal for three rounds. Starting from the SAT solution which had wirelength 472 [see Fig. 12(a)], the first round decreases it to 380, the second round decreases it to 364 and the third and last round sets it to 360 [see Fig. 12(b)]. Thus, a 23% reduction of wirelength is achieved. It is clear that the first two reduction rounds have a substantial impact on wirelength. The impact of the third round is more dubious. With respect to the running time, the SAT phase took about

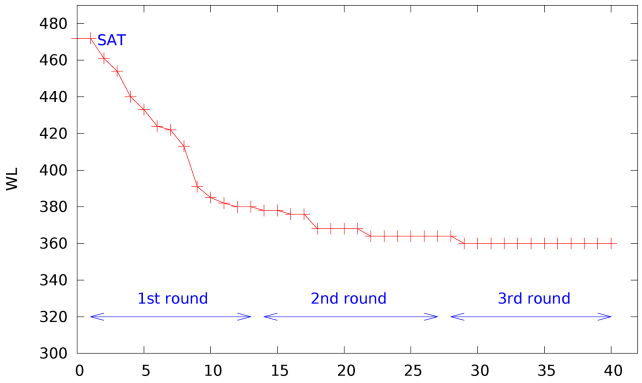


Fig. 11. Wirelength reduction from first SAT solution.

half a minute to find a satisfiable assignment for the whole program, while each ILP&LNS iteration for an individual signal took beetween 0.3 and 1 s, for a total of 16 s. This is explained by the fact that the SAT problem has about 40 000 variables, while each ILP&LNS model has 1350 variables on average. On the other hand, the complete minimization problem (all nets simultaneously) could not be solved in 12 hours.

E. Routing of Single-Height Cells for Complete Library

The layouts for single-height cells were generated for the complete Nangate 45nm Library. All cells were routed using $\mathcal{W} = 6$. The total CPU time to route without optimizing all the cells was about 45 minutes. The longest one (`DFFR_X1`) took 9 min.

In order to reduce the wirelength, two optimization rounds were performed, reducing the wirelength of one net at a time. More rounds did not reduce the wirelength substantially. On average, wirelength was reduced by about 35% at the expense of doubling the total CPU time (93 min). This time was distributed as follows: 41% of the time was spent in solving the SAT models to find a legal routing, 49% applying the ILP&LNS heuristic for wirelength reduction, and the rest in generating and simplifying the CNF formulas before feeding them to the solvers.

Individual data and layouts for each cell can be found in the link "`Single-height cells`" of the website.

F. Synthesis of Double-Height Cells

Similarly, double-height layouts were generated for the 37 largest cells in the library (those having, at least, seven polysilicon columns). The same wirelength optimization process than before was applied but setting the windowing parameter to $\mathcal{W} = 8$.

All but three original double-height cells could be routed; let us analyze the cases that could not be directly routed.

1) On one hand, the minimum placements of the `DFFR_X2` and `SDFFR_X2` cells were so congested, that they could never be routed. To solve this issue, we substantially reduced their congestion by increasing their area by a polysylicon column, and could then route them normally.

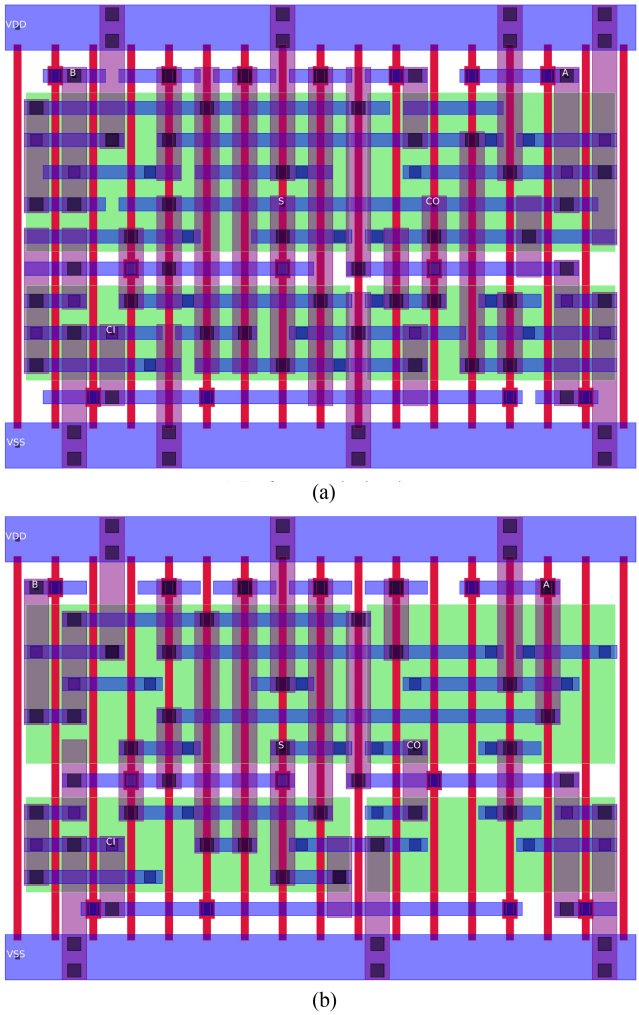


Fig. 12. FA_X1 before and after wirelength optimization. (a) Before optimization. (b) After optimization.

2) On the other hand, no routing of the `FA_X1` cell could be found in less than 12 hours, even though its congestion is moderate. In this case, we allowed the router to use the pyramidal ceiling for just one signal (as described in Section IV-D), and a valid routing was delivered in half a minute.

Taking into account the previous items, the total CPU time to route (without wirelength optimization) this subset of cells was 112 min. The longest one (`DFFRS_X2`) took 48 min. The total CPU time to route and reduce the wirelength was 2 hours and 35 min. One can observe that double-height cells require more effort from the SAT solver than single height cells.

Individual data and layouts for each cell can be found in the link "`Double-height cells`" of the website.

G. Commercial Cell Library

In order to evaluate the quality of the results obtained by the algorithms presented in this paper, we compared the layouts of a commercial library with the one derived by our tool. The comparison was done for a low-power 10-track library with 206 cells. The layouts were using 1D geometries for poly,

TABLE V
RESULTS FROM A COMMERCIAL LIBRARY

Layer	Wirelength (nm)		Δ
	Original	Automatic	
M1H	1170360	1116450	-4.6%
M1V	235960	217940	-7.6%
M2V	215900	273530	+26.7%
M3H	25380	12510	-50.7%
Total	**1647600**	**1620430**	**-1.6%**
Cells using M3	8	2	
Vias M1-M2	662	997	+50.6%
Vias M2-M3	44	20	-55.5%

M2 and M3, and 2-D geometries for M1. M3 was only used occasionally to route complex cells. In this library, the FEOL geometries (poly and diffusion) were always aligned with a uniform grid.

The experiment was performed as follows.

1) The FEOL geometries and the coordinates of the input/output terminals were extracted from the GDSII file. Therefore, the area of the standard cells was not modified.
2) A set of gridded design rules was created (about 30 design rules) to satisfy the ungridded design rules defined by the technology.
3) The cells were routed on the same grid defined for the FEOL layers. M3 was only used when the cell was unroutable with only two metal layers.
4) DRC was run on the final layouts to check conformance with the original (ungridded) design rules.
5) The wirelength for every metal layer and the number of vias were calculated for the original and the newly created cells.

Table V reports a summary of the results for the original library and the one obtained by our routing tool (automatic). The first important observation is that the original library used M3 for eight cells, whereas our tool only required M3 for two cells. The eight cells (full/half adders and some flip-flops) showed highly congested layouts that were difficult to complete with only two metal layers.

The layouts produced by our tool had a total wirelength reduction of 1.6% with regard to the original layouts. We can conclude that the tool can derive very competitive results, even with the constraint of using gridded layouts.

The main reduction produced by our tool was in horizontal and vertical M1 (M1H and M1V), but this reduction was compensated by an increase of vertical M2 (M2V), which also results in an increase of M1-M2 vias. The reduction in M3 and M2-M3 vias is due to the fact that only two cells required the M3 for the completion of routing.

VIII. CONCLUSION

Regular fabrics open a new avenue of opportunities for physical design automation. This paper has presented an approach to solve the problem of detailed routing for the synthesis of library cells in which parameterization and technology independence are key aspects.

The discretization of the cell routing problem and the embedding into a grid graph enables the modeling as a combinatorial problem for which powerful heuristics can be used to solve it. The algorithmic approach presented in this paper has been shown to be versatile and competitive.

Current SAT solvers have powerful resolution engines and the problem encoding proposed in this paper has contributed to reduce the CPU time by few orders of magnitude with regard to previous encoding schemes. However, the lack of information about the nature of the problem often lets the SAT solver to do an unguided exploration of infertile branches of the tree. We believe that a guided selection of the branching variables may contribute to speed-up even more the algorithm.

Performance, power and area are interrelated factors that need to be tradedoff in a different way for each particular design. The availability of this type of EDA tools will enable the exploration and characterization of a wide range of layout templates for cell libraries. This is one of the directions for future work.

REFERENCES

[1] R. F. Pease and S. Y. Chou, "Lithography and other patterning techniques for future electronics," *Proc. IEEE*, vol. 96, no. 2, pp. 248–270, Feb. 2008.
[2] J. C. Poirier, "Excellerator: Custom CMOS leaf cell layout generator," *IEEE Trans. Comput. Aided Design*, vol. 8, no. 7, pp. 744–755, Jul. 1989.
[3] M. Guruswamy, R. L. Maziasz, D. Dulitz, S. Raman, V. Chiluvuri, A. Fernandez, *et al.*, "CELLERITY: A fully automatic layout synthesis system for standard cell libraries," in *Proc. ACM/IEEE Design Autom. Conf.*, 1997, pp. 327–332.
[4] W. Maly, Y.-W. Li, and M. Marek-Sadowska, "OPC-free and minimally irregular IC design style," in *Proc. ACM/IEEE Design Autom. Conf.*, 2007, pp. 954–957.
[5] N. Ryzhenko and S. Burns, "Physical synthesis onto a layout fabric with regular diffusion and polysilicon geometries," in *Proc. ACM/IEEE Design Autom. Conf.*, 2011, pp. 83–88.
[6] V. Kheterpal, V. Rovner, T. Hersan, D. Motiani, Y. Takegawa, A. J. Strojwas, *et al.*, "Design methodology for IC manufacturability based on regular logic-bricks," in *Proc. ACM/IEEE Design Autom. Conf.*, 2005, pp. 353–358.
[7] B. Taylor and L. Pileggi, "Exact combinatorial optimization methods for physical design of regular logic bricks," in *Proc. ACM/IEEE Design Autom. Conf.*, 2007, pp. 344–349.
[8] W. Hung, X. Song, T. Kam, L. Cheng, and G. Yang, "Routability checking for three-dimensional architectures," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 12, no. 12, pp. 1371–1374, Dec. 2004.
[9] S. Wimer, R. Y. Pinter, and J. A. Feldman, "Optimal chaining of CMOS transistors in a functional cell," *IEEE Trans. Comput. Aided Design*, vol. 6, no. 5, pp. 795–801, Sep. 1987.
[10] C.-Y. Hwang, Y.-C. Hsieh, Y.-L. Lin, and Y.-C. Hsu, "A fast transistor-chaining algorithm for CMOS cell layout," *IEEE Trans. Comput. Aided Design*, vol. 9, no. 7, pp. 781–786, Jul. 1990.
[11] F. Yang, Y. Cai, Q. Zhou, and J. Hu, "SAT based multi-net rip-up-and-reroute for manufacturing hotspot removal," in *Proc. DATE*, Mar. 2010, pp. 1369–1372.
[12] L.-D. Huang and M. Wong, "Optical proximity correction (OPC)-friendly maze routing," in *Proc. ACM/IEEE Design Autom. Conf.*, Jul. 2004, pp. 186–191.
[13] J. Mitra, P. Yu, and D. Z. Pan, "RADAR: RET-aware detailed routing using fast lithography simulations," in *Proc. ACM/IEEE Design Autom. Conf.*, Jun. 2005, pp. 369–372.
[14] H. Yao, Y. Cai, and X. Hong, "CMP-aware maze routing algorithm for yield enhancement," in *Proc. IEEE Comput. Soc. Annu. Symp. Very Large Scale Integr. Syst.*, Mar. 2007, pp. 239–244.
[15] N. Ryzhenko and S. Burns, "Standard cell routing via Boolean satisfiability," in *Proc. ACM/IEEE Design Autom. Conf.*, 2012, pp. 603–612.
[16] J. Hu and S. Sapatnekar, "A survey on multi-net global routing for integrated circuits," *Integr. VLSI J.*, vol. 31, no. 1, pp. 1–49, 2001.
[17] R. Carden, J. Li, and C.-K. Cheng, "A global router with a theoretical bound on the optimal solution," *IEEE Trans. Comput. Aided Design*, vol. 15, no. 2, pp. 208–216, Feb. 1996.

[18] Y.-W. Lin, M. Marek-Sadowska, and W. Maly, "Layout generator for transistor-level high-density regular circuits," *IEEE Trans. Comput. Aided Design*, vol. 29, no. 2, pp. 197–210, Feb. 2010.
[19] M. Lavin, F.-L. Heng, and G. Northrop, "Backend CAD flows for 'restrictive design rules'," in *Proc. ICCAD*, 2004, pp. 739–746.
[20] D. Abercrombie, P. Elakkumanan, and L. Liebmann, "Restrictive design rules and their impact on 22 nm design and physical verification," in *Proc. EDPS*, 2009.
[21] A. Kahng, C.-H. Park, X. Xu, and H. Yao, "Layout decomposition for double patterning lithography," in *Proc. ICCAD*, 2008, pp. 465–472.
[22] B. Taylor, "Automated layout of regular fabric bricks," Master's thesis, Dept. of Elect. Comput. Eng., Carnegie Mellon University, Pittsburgh, PA, USA, Dec. 2005.
[23] S. Prestwich, "CNF encodings," in *Handbook of Satisfiability*, A. Biere, M. Heule, H. v. Maaren, and T. Walsh, Eds. Amsterdam, The Netherlands: IOS Press, 2009, pp. 75–97.
[24] M. Davis and H. Putnam, "A computing procedure for quantification theory," *J. ACM*, vol. 7, no. 3, pp. 201–215, 1960.
[25] L. Euler, "Solutio problematis ad geometriam situs pertinentis," *Commentarii academiae scientiarum Petropolitanae*, vol. 8, pp. 128–140, 1741.
[26] N. Eén and N. Sörensson, "Translating pseudo-Boolean constraints into SAT," *J. Satisfiability, Boolean Model. Comput.*, vol. 2, pp. 1–26, 2006.
[27] G. R. Raidl and J. Puchinger, "Combining (integer) linear programming techniques and metaheuristics for combinatorial optimization," in *Hybrid Metaheuristics (Studies in Computational Intelligence)*, vol. 114, C. Blum, M. Blesa, A. Roli, and M. Sampels, Eds. Springer-Verlag Berlin Heidelberg, 2008, pp. 31–62.
[28] R. K. Ahuja, Ö. Ergun, J. B. Orlin, and A. P. Punnen, "A survey of very large-scale neighborhood search techniques," *Discrete Appl. Math.*, vol. 123, nos. 1–3, pp. 75–102, 2002.
[29] J. Puchinger, G. R. Raidl, and U. Pferschy, "The core concept for the multidimensional knapsack problem," in *Proc. Conf. Evol. Comput. Combinatorial Optimiz.*, LNCS 3906. 2006, pp. 195–208.
[30] A. Biere. *PicoSAT* [Online]. Available: http://fmv.jku.at/picosat
[31] Gurobi Optimization, Inc. (2012). *Gurobi Optimizer Reference Manual* [Online]. Available: http://www.gurobi.com
[32] S. Minato, "Fast generation of prime-irredundant covers from binary decision diagrams," *IEICE Trans. Fund. Electron. Commun. Comput. Sci.*, vol. E76-A, no. 6, pp. 967–973, Jun. 1993.
[33] F. Somenzi. *CUDD: CU Decision Diagram Package* [Online]. Available: http://vlsi.colorado.edu
[34] *NanGate 45nm Open Cell Library* [Online]. Available: http://nangate.com
[35] M. Smayling, "Gridded design rules: 1-D approach enables scaling of CMOS logic," *Nanochip Technol. J.*, vol. 6, no. 2, pp. 33–37, 2008.
[36] R. Bar-Yehuda, J. A. Feldman, R. Y. Pinter, and S. Wimer, "Depth-first-search dynamic programming algorithms for efficient CMOS cell generation," *IEEE Trans. Comput. Aided Design*, vol. 8, no. 7, pp. 737–743, Jul. 1989.
[37] C. Gomes, H. Kautz, A. Sabharwal, and B. Selman, "Chapter 2: Satisfiability solvers," in *Handbook of Knowledge Representation (Foundations of Artificial Intelligence)*, vol. 3, F. Harmelen, V. Lifschitz, and B. Porter, Eds. Amsterdam, The Netherlands: Elsevier, 2008, pp. 89–134.

Jordi Cortadella (M'88) received the M.S. and Ph.D. degrees in computer science from the Universitat Politècnica de Catalunya, Barcelona, Spain, in 1985 and 1987, respectively.

He is currently a Professor with the Department of Software, Universitat Politècnica de Catalunya. In 1988, he was a Visiting Scholar with the University of California, Berkeley, CA, USA. He has co-authored numerous research papers and has been invited to present tutorials at various conferences. His current research interests include formal methods and computer-aided design of very large scale integration systems with a special emphasis on asynchronous circuits, concurrent systems, and logic synthesis.

Dr. Cortadella has served on technical committees of several international conferences in the fields of design automation and concurrent systems. He received the Best Paper Awards at the International Symposium on Advanced Research in Asynchronous Circuits and Systems in 2004, the Design Automation Conference in 2004, and the International Conference on Application of Concurrency to System Design in 2009. In 2003, he was the recipient of a Distinction for the Promotion of the University Research by the Generalitat de Catalunya. He is a member of the Academia Europaea.

Jordi Petit received the M.S. degree in computing engineering and the Ph.D. degree in computer science from the Universitat Politècnica de Catalunya, Barcelona, Spain, in 1996 and 2001, respectively.

Since 2002, he has been an Associate Professor with the Facultat d'Informàtica de Barcelona, Barcelona, and with the Facultat de Matemàtiques i Estadística, within the Departament de Llenguatges i Sistemes Informàtics. He has co-authored numerous research papers published in peer-reviewed journals and international conferences. His current research interests include algorithm engineering. He has worked on topics such as the design, analysis, and implementation of algorithms and data structures (in sequential, distributed, and parallel settings), heuristic approaches to combinatorial optimization problems (e.g., on layout and routing), and the use of probabilistic models to study fundamental aspects of sensor networks. Most of his work tries to bridge the gap between theory and practice and involves development of usable software for various state-of-the-art infrastructures.

Sergio Gómez received the M.S. degree in telecommunication engineering from the Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, in 2009. He is currently pursuing the Ph.D. degree with the Department of Electronic Engineering, UPC.

He spent one year researching wireless mesh networks at King's College, London, U.K. His current research interests include very large scale integration design, design for manufacturability, lithography enhanced layout design, and litho-friendly standard cell library creation.

Francesc Moll (M'92) received the M.S. degree in physics from the Universitat de les Illes Balears, Palma, Spain, and the Ph.D. degree in electronic technology from the Universitat Politècnica de Catalunya (UPC), Barcelona, Spain.

He has been a Professor of electronic technology with the Department of Electronic Engineering, UPC, since 1993. He is a member of the Research Group HIPICS. He has published numerous papers in international conferences and journals. His current research interests include very large scale integration design and test, manufacturing variations, and defect modeling in integrated circuits, defect and fault tolerant systems, resilient mechanisms, low-power nanoelectronics, and energy harvesting systems.

Dr. Moll serves on the technical committees of several international conferences in the fields of integrated circuit design and as a reviewer in several international journals.