IPSJ Transactions on System LSI Design Methodology Vol.9 2–12 (Feb. 2016)

[DOI: 10.2197/ipsjtsldm.9.2]

Invited Paper

Placement: From Wirelength to Detailed Routability

Wing-Kai Chow[1,a)] Evangeline F.Y. Young[1,b)]

Received: December 15, 2015

Abstract: Research on the placement problem in physical design has evolved timely in the recent few decades from traditional wirelength-driven, to routability-driven and then to detailed-routability driven. In this paper, we will focus on the interconnect and routing issues in placement, and study and survey on the development and progress of related works on this important problem.

Keywords: Placement, routability, physical design

1. Introduction

Placement is a classical problem in VLSI physical design and is also an important step that will directly affect and determine the performance of a circuit design. Given a floorplan, a set of cells and a netlist, the goal of placement is to assign a location and an orientation to every movable cell satisfying constraints (such as non-overlapping) and meeting the objectives (such as wirelength and timing). For large circuits with millions of cells, the placement process is usually divided into three stages: *global placement*, *detailed placement*, and *legalization*. In global placement, some details of the cells such as orientation and cell alignment issues are usually ignored. Global placement targets at producing a rough placement solution with minimized objective function (e.g., total wirelength) and reasonable density distribution. Legalization aims at removing cell overlapping by adjusting the cell position minimally. In detailed placement, cells are usually moved or swapped locally to further improve the placement objective.

The early placement works in or before 1990's are wirelength-driven. Although minimization of total wirelength implies minimization of total routing demand, it does not reflect the existence of local routing congestion. As technology advances, shrinking of feature size increases pin density significantly. Placement without considering routing usually results in serious routing congestion problem due to an uneven distribution of the routing demands. Various congestion estimation and routability-driven placement methods appear since 2000's. In 2010's, the focus started to shift towards detailed-routability issues. With the fact that the number of design rules has increased exponentially in the recent decade, placement can no longer ignore the detailed routing issues. So far a few works addressing the detailed routability have been proposed.

We can see different generations of placers and placement techniques in the past 20 years. However, the problem is still far from being solved because of the forever increasing complexity and functionality of circuit designs and the consequent placement and layout constraints. In addition, placement is a multi-objective optimization problem that finally needs to make use of some routers to determine whether the objectives are fulfilled, which makes the problem even harder to tackle.

Wirelength-driven placement is relatively simpler comparing with the other two objectives of congestion and detailed routability because there is a simple metric, half-perimeter wirelength (HPWL) that was found to correlate with the routing wirelength to an extent [1]. With this HPWL modeling, some good techniques can be devised for global placement, legalization and detailed placement. The wirelength-driven placement problem has been well studied and researched. Once moving into the regime of routability, which is the real concern of placement, the complexity of the problem increases while our understanding of the methodology is not much. Today, one of the key challenges in modern physical synthesis is routability, due to several factors like increased use of embedded IPs and memories on die that block metal layers for routing, smaller die size to control manufacturing cost and complicated logic structures. There are efforts in modeling congestion and routability with some probabilistic metrics, but there is not yet any simple metric that is shown to be correlated directly to the final routability. Routability-driven placement becomes a hot topic in recent years and a series of routability-driven placement contests were organized in different conferences (ISPD 2011 [2], DAC 2012 [3], ICCAD 2012 [4], ISPD 2014 [5], ISPD2015 [6]) to promote research in this field.

Moving next into detailed routability-driven placement makes the problem even more complex, since we now need to consider the exact geometry of the routing wires. Layout details like power/ground alignment, pin access and design rules like minimum wire spacing are essential issues to be considered. There are not many works reported on this problem and most of the techniques are greedy and heuristic based. In this paper, we will review on those representative placement methods and techniques for different placement objectives and constraints.

1 Department of Computer Science and Engineering, Chinese University of Hong Kong, Shatin, Hong Kong

a) wkchow@cse.cuhk.edu.hk

b) fyyoung@cse.cuhk.edu.hk

In this paper, we review and discuss some of the techniques of wirelength-driven placement in Section 2, in which method of wirelength estimation (Section 2.1), various placement frameworks of global placement (Section 2.2, 2.3, 2.4), legalization (Section 2.5), detailed placement (Section 2.6) will be discussed. In Section 3, issues of routability-driven placement will be reviewed, in which methods of estimating routing congestion (Section 3.1) and techniques of removing the congestion (Section 3.2) will be introduced. In Section 4, the handling of detailed routing issues in placement will be presented. Some of the important design rules are discussed in Section 4.1 and works on handling of such design rules are introduced in Section 4.2. Section 5 concludes our paper.

2. Wirelength-driven Placement Basics

2.1 Wirelength Estimation

During the placement stage, the interconnection of the cells are given as netlist. For each net, a list of pins (on the cells), or fixed pins (on the floorplan or fixed cells) are given. However, how they are connected with metal wires cannot be determined at this stage until placement and routing are done. Furthermore, the routing topology and thus the actual wirelength depend on the choice of the router being used. Routers have various routing strategies and can produce very different results with different wirelengths. In order to evaluate the quality of a placement without knowing any information about routing, a number of ways to estimate the length of a net given only the positions of the pins in a net are commonly used.

2.1.1 Half-perimeter Wirelength (HPWL)

Half-Perimeter Wirelength (HPWL) of a net is the sum of the width and the height of the minimal bounding box containing every pin of the net.

$$HPWL = \max_{i,j\in N} |x_i - x_j| + \max_{i,j\in N} |y_i - y_j| \qquad (1)$$

It can be calculated efficiently in $O(n)$ time where n is the total number of pins. HPWL is the minimum rectilinear wirelength required to connect every pin. Without considering routing resource, HPWL is accurate for two- and three-pin nets. However, it usually under-estimates the wirelength of multi-pin nets.

2.1.2 Rectilinear Minimum Spanning Tree (RMST)

In rectilinear minimum spanning tree (RMST), the distance between each pair of pins is assumed to be their shortest Manhattan distance. The RMST is then built by constructing a minimum spanning tree among all the pins. The wirelength estimated with RMST is more accurate than that with HPWL. The computational time is $O(n \log n)$ for each net where n is the number of pins in the net and is usually affordable. Since every edge in an RMST tree is pin-to-pin, with appropriate edge flipping, some edges may be shared by multiple pin-to-pin connections. Therefore, RMST will very often over-estimate the routing wirelength.

2.1.3 Rectilinear Steiner Minimal Tree (RSMT)

Rectilinear Steiner minimal tree (RSMT) wirelength is the actual minimum rectilinear wirelength required to connect all the pins of a net. When routing resource is not limited, it is considered as the most accurate wirelength estimation. However, construction of RSMT is NP-hard, and the computational time to find the exact solution is too long.

In practice, all of the above wirelength estimations are too optimistic due to the following reasons:

(1) There are routing blockages due to pre-placed multi-layer macros or pre-routed nets.

(2) Routing resource is not unlimited. When multiple nets are routed in a local region, the wires of the nets will block each other.

(3) Routing is usually done on multiple uni-directional routing layers connected with vias. Vias are resistive and their usages should be minimized. However, the net topology computed by the above wirelength estimation methods do not consider any via usage or routing strategy and thus will bring inaccuracy.

2.2 Partitioning-based Placement

Due to the huge circuit size, a popular approach is to break down the input circuit into multiple smaller and manageable sub-circuits. In partitioning-based placement, the netlist is divided into smaller sub-netlists and the layout is divided into smaller sub-regions. Each sub-netlist is assigned to a sub-region and the placement of each sub-netlists are handled separately. As the problem size of each sub-region is smaller, the placement can be done in much lower runtime.

2.2.1 Hypergraph Partitioning Algorithm

An enabling technique for partitioning-based/min-cut placement is an effective hypergraph partitioning. Finding the optimal bisection of a hypergraph is NP-complete. Alpert et al. [7] did a survey on pre-90's researches on hypergraph partitioning. Among them, hMetis [8] and BestChoice clustering [9] are commonly used partitioning algorithm for min-cut placement [10]. hMetis [8] performs partitioning by a hierarchically clustering and decluttering the cells. The work of Ref. [11] proposed a spectral partitioning heuristic that uses eigenvectors to construct a geometric embedding of a given graph which is then partitioned. BestChoice clustering [9] proposed an iterative clustering technique based on a score function:

$$d_c(u,v) = \sum_{N:u,v\in N} \frac{1}{|N|} \cdot \frac{1}{a(u)+a(v)} \qquad (2)$$

where u and v are the two clusters of cells, N is a net connecting both u and v, $|N|$ is the total number of cells in net N, and $a(x)$ is the total area of the cells inside cluster x. The clustering strategy will combine clusters with high scores.

BonnPlace [10] modified BestChoice clustering with consideration of cell locations. Sizes of the bounding boxes containing the cell locations are considered in the clustering cost such that cells geometrically closer are more likely to be clustered. POLAR 3.0 [12] uses *placement-driven partitioning*, in which the partitioning is purely based on cells' locations. Starting with a roughly legalized initial placement, the circuit is partitioned with horizontal and vertical cuts such that the number of cells in each partition are similar. During placement, all the partitions are placed in parallel independently assuming that all the pins in the other partitions are fixed.

2.3 Placement with Lower/Upper Bound Framework

There are a number of placers developed based on an iterative lower and upper bound framework [13], [14], [15], [16]. The whole global placement process iterates between a lower bound computation, which optimizes the wirelength, and an upper bound computation, which moves cells apart to remove overlap. The two processes will gradually converge by different means of incorporating the solutions of one process into the other.

The idea of iteratively coordinating wirelength-optimized placement and overlap-free placement has been proposed in many previous works [17], [18], [19]. For example, Kraftwerk [19] solves a quadratic formulation to minimize wirelength to give an initial placement solution. In global placement, the cells are spread iteratively over the chip. *Potential* is defined at each location as the local difference between cell area and free area (also called local cell area overflow). The gradients of the potential determine a move force for each cell to reduce cell overlapping. On the other hand, a hold force is defined to oppose a net force which is defined to reduce wirelength to retain a cell at its overlap-free location obtained from the last iteration.

2.3.1 Lower Bound—Quadratic Placement

In quadratic placement, the interconnection of cells are modeled in such a way that every pair of cells are associated with a zero net weight when they are not connected, while they may or may not be associated with a non-zero net weight when they are connected, depending on the net model in use. In another word, the interconnections are represented as a set of two-pin nets, where a net weight is associated with each net.

(a) Interconnect Model. In the formulation of quadratic placement, two-pin nets are handled accurately. To consider multi-pin nets, we need to convert the multi-pin nets into a set of weighted two-pin nets such that the wirelength of the multi-pin net will correspond to the the total weighted wirelength of the set of two-pin nets.

The *clique model* is the simplest formulation, in which all the pins in a net are connected to each other to form a complete graph (**Fig. 1** (a)). The resulting $k(k-1)/2$ edges for a k-pin net may be too many especially for nets with high fan-in and fan-out. To compensate for the excessive connections in the clique model, the net weight is set to $2/k$. Clique model were used in Refs. [20], [21]. In *star model*, an extra dummy node (the star) is created and every pin in the net is connected to it (Fig. 1 (b)). Therefore, there are n edges and $(n+1)$ nodes in the star model. The model is good for considering timing as it captures the signal flow from the output pin of a gate to the input pins of the other gates. It is proved in Ref. [22] that, with appropriate net weights and when the net force acting on the star node is zero, the clique model is equivalent to the star model in quadratic placement. Both net models can give the similar solution quality, but they introduce different number of nodes and edges under different number of pins of the net. Therefore, the work of Ref. [22] uses combination of the clique model and the star model, in which the clique model is used for two- or three-pin nets, while the star model is used for nets with more than three pins.

Bound-2-Bound (B2B) Model is proposed by Ref. [19]. In the x-direction (or y-direction), the boundary pins with the maximum and the minimum coordinate values are identified and all other pins of the net are assummed to be connecting to these two boundary pins (Fig. 1 (a), (b)). A drawback is that in this model, the connections between pins in a net depend on the pins' positions. Once the placement is changed, the pins with the maximum and minimum coordinates may change and the connection between the pins will need to be changed. Many recent analytical placers [12], [13], [23] use this B2B model, because of its resemblance to the HPWL.

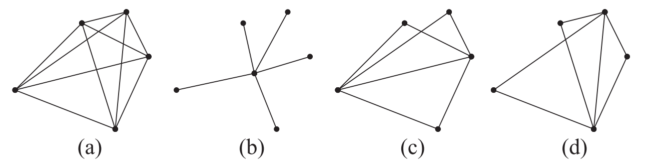

Fig. 1 Wire models for quadratic placement. (a) Clique model (b) Star model (c) B2B model in x-direction (d) B2B model in y-direction.

(b) Objective Function. The objective function of quadratic placement is the sum of the squared wirelength of each two-pin connection, i.e.,

$$\Phi(x, y) = \sum_{i,j \in C, i<j} w^x_{i,j}(x_i - x_j)^2 + w^y_{i,j}(y_i - y_j)^2 \tag{3}$$

where x and y are the vectors of the x- and y-coordinates of all the cells, $w^x_{i,j}$ and $w^y_{i,j}$ are the weights of the connections between pin i and pin j, which is zero when the pins are not connected in the net model (but they may be connected in the actual netlist). The weights of a two-pin net may be different in different direction under some net models, e.g., in the B2B model, two cells may be connected in one direction, but not in the another direction. It is easy to see that the objective function can be separated into the x- and y-direction.

$$\Phi(x) = \sum_{i,j \in C, i<j} w^x_{i,j}(x_i - x_j)^2 \tag{4}$$

A Laplacian matrix Q_x is used to represent the graph of all two-pin connections. The above objective function can then be rewritten as:

$$\Phi(x) = \frac{1}{2} x^T Q_x x + d_x^T x + constant \tag{5}$$

where Q_x is the $n \times n$ Laplacian matrix for all the two-pin nets, d_x is a vector of constants with size n. The above objective function is minimized by solving the linear system:

$$Q_x x + d_x = 0 \tag{6}$$

To solve the linear system, some commonly used methods are conjugate gradient (CG) descent and successive over-relaxation (SOR).

Although the wirelength can be minimized effectively with the quadratic placement method, the placement result is not realistic due to excessive cell overlapping. To solve the overlapping problem, some common approaches are implicitly integrating into the objective a function representing the extent of cell overlap, or explicitly spreading cells away from each other while preserving the wirelength.

2.3.2 Upper Bound

The purpose of upper bound is to remove cell overlapping. Most works evaluate the amount of overlapping as follow. The whole placement region is partitioned into a set of regular rectangular bins. For each bin, free area and cell area are calculated. Free area is the area that can be used to accommodate cells, while cell area is the total area of the cells inside the bin.

SimPL [13], Ripple [14], POLAR [16], and EhPlacer [15] use a similar approach of partitioning the chip into a set of regular rectangular bins. The amount of cell area overflow is defined as the difference between cell area and free area. Overflowed regions are identified as groups of bins with cell area overflow. Windows are defined centering at the overflowed region. The window's size is adjusted such that it is the minimum size with the cell area equal the free area. Cells are then spread out linearly in the horizontal and vertical direction to even out the distribution of the cells, while the relative order of the cells inside the window are maintained. After cell spreading, the cells will not overlap but the wirelength will increase. Another round of wirelength minimization will then be performed taking into account the non-overlap positions of the cells obtained after this cell spreading process. This is achieved with the introduction of weighted pseudo-nets that connect each cell to its location after cell spreading.

2.4 Analytical Non-linear Placement

In non-linear placement, the objective function is formulated as a non-linear function and being optimized with some analytical programming techniques. This approach usually provides higher flexibility in solving multi-objective problems. For example, both wirelength and local cell density (thus reducing overlap) can be optimized simultaneously. In order to solve the problem effectively, the objectives will be formulated as smooth mathematical functions.

2.4.1 Wirelength Model

The HPWL formulation is neither smooth nor convex. There are efforts to approximate HPWL with a curve function:

Log-Sum-Exp (LSE) wirelength model is proposed by Naylor et al. [24]. Instead of converting a hypernet into a set of two-pin nets, the HPWL function is smoothed as log of sum of exponentials of the coordinates as follows:

$$\gamma\left(\ln\sum_{i\in N} e^{\frac{x_i}{\gamma}} + \ln\sum_{i\in N} e^{-\frac{x_i}{\gamma}}\right) + \gamma\left(\ln\sum_{i\in N} e^{\frac{y_i}{\gamma}} + \ln\sum_{i\in N} e^{-\frac{y_i}{\gamma}}\right) \quad (7)$$

where N is the set of all pins in a net, (x_i, y_i) is the position of pin i, and γ is a constant close to zero. When γ approaches zero, the above formulation equals HPWL. However, to avoid arithmetic overflow in computation, γ cannot be too small. In NTUPlace3 [25], γ was chosen to be 1% of the chip width.

Weighted-Average (WA) wirelength model is proposed by Ref. [26] that approximates the max and min term in HPWL:

$$\frac{\sum_{i\in N} x_i e^{\frac{x_i}{\gamma}}}{\sum_{i\in N} e^{\frac{x_i}{\gamma}}} - \frac{\sum_{i\in N} x_i e^{-\frac{x_i}{\gamma}}}{\sum_{i\in N} e^{-\frac{x_i}{\gamma}}} + \frac{\sum_{i\in N} y_i e^{\frac{y_i}{\gamma}}}{\sum_{i\in N} e^{\frac{y_i}{\gamma}}} - \frac{\sum_{i\in N} y_i e^{-\frac{y_i}{\gamma}}}{\sum_{i\in N} e^{-\frac{y_i}{\gamma}}} \quad (8)$$

When γ approaches zero, the model converges to HPWL. It can produce smaller estimation errors than the Log-Sum-Exp model.

Recently, the wirelength model proposed by Ray and Balachandran [27] can give an even lower bound of error:

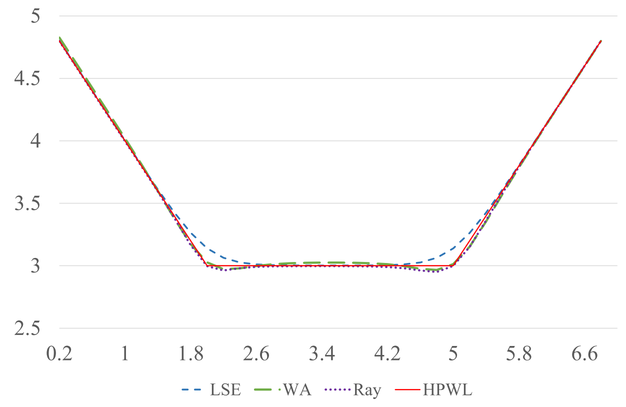

Fig. 2 Comparison of different HPWL smoothing wirelength models. The graph shows a net with pins at x-coordinates = $\{2, 4, 5, x\}$, y-axis is the resulting wirelength with different x under different wirelength models.

$$\frac{\sum_{i\in N} x_i^p e^{\frac{x_i}{\gamma}}}{\sum_{i\in N} x_i^{p-1} e^{\frac{x_i}{\gamma}}} - \frac{\sum_{i\in N} x_i^{-p} e^{\frac{x_i}{-\gamma}}}{\sum_{i\in N} x_i^{-p-1} e^{\frac{x_i}{-\gamma}}} + \frac{\sum_{i\in N} x_i^p e^{\frac{y_i}{\gamma}}}{\sum_{i\in N} x_i^{p-1} e^{\frac{y_i}{\gamma}}} - \frac{\sum_{i\in N} x_i^{-p} e^{\frac{y_i}{-\gamma}}}{\sum_{i\in N} x_i^{-p-1} e^{\frac{y_i}{-\gamma}}} \quad (9)$$

When γ approaches zero and p approaches infinity, the model converges to HPWL. **Figure 2** compares the accuracy of the three models. We can observe that all three net models successfully smooth the HPWL function, while WA model and Ref. [27]'s model give significantly better approximations of HPWL than that of LSE model.

2.4.2 Overlap Reduction

To reduce overlap during placement, the amount of cell overlapping is also integrated into the objective function in APlace [28]:

$$P(c,g) = K_c \cdot p(|x_c - x_g|) \cdot p(|y_c - y_g|)$$
$$p(d) = \begin{cases} 1 - \frac{2d^2}{r^2} & 0 \le d \le \frac{r}{2} \\ \frac{2(d-r)^2}{r^2} & \frac{r}{2} < d \le r \\ 0 & r < d \end{cases} \quad (10)$$

where $p(d)$ is a bell-shaped potential function representing the amount of overlap as a function of the distance between the cell center and the grid center, and K_c is a normalization factor so that the total sum of potential of a cell equals to its area. **Figure 3** compares the potential function and the actual amount of overlap.

With consideration of overlap minimization, the objective function now becomes:

$$\min \alpha W(x,y) + \beta \sum_{g\in G}\left(\sum_{c\in C} P(c,g) - P_{ex}(g)\right)^2 \quad (11)$$

where $W(x,y)$ is the total wirelength, G is the set of all grids, C is the set of all cells, $P_{ex}(g)$ is the expected potential of grid g (the potential of g when cells are evenly distributed on the chip). The parameter α is the weight for wirelength and β is the weight for density satisfaction.

As APlace applies the same potential function to every cell, it does not consider the effect of cell sizes on the overlapping. In fact, a large cell will start overlap with a grid at larger distance

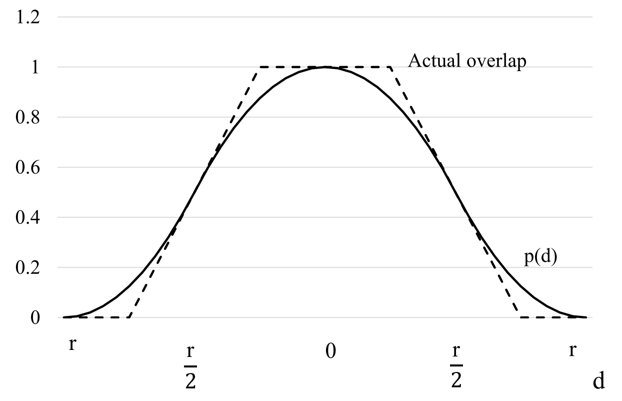

Fig. 3 Bell-shaped potential function representing the amount of overlap as a function of the distance d between the cell center and the grid center.

away from the grid. NTUPlace [25] thus modifies the potential function to consider cell size and grid size. For example, in the x-direction, the potential function $p_x(c, g)$ is:

$$P(c, g) = K_c \cdot p_x(c, g) \cdot p_y(c, g)$$

$$p_x(c, g) = \begin{cases} 1 - \frac{4}{(w_c+2w_b)+(w_c+4w_g)} d_x^2 & 0 \le d_x \le \frac{w_c}{2} + w_g \\ \frac{2}{w_g(w_c+4w_g)} (d_x - \frac{w_c}{2} - 2w_g)^2 & \frac{w_c}{2} + w_g < d_x \le \frac{w_c}{2} + w_g \\ 0 & \frac{w_c}{2} < d_x \end{cases} \tag{12}$$

where w_c is the cell width and w_g is the grid width. $p_y(c, g)$ is defined similarly regarding the cell height. With consideration of overlap minimization, the objective function for NTUPlace becomes:

$$\min W(x, y) + \lambda \sum_{g \in G} \left(\sum_{c \in C} P(c, g)) - P_{ex}(g) \right)^2 \tag{13}$$

where λ is the penalty factor for violation of expected local potential. The non-linear objective function is solved by the non-linear conjugate gradient (CG) method with increasing λ.

2.5 Legalization

Tetris [29] is a greedy method to legalize every cell in almost a negligible amount of time. Cells are sorted by their x-coordinates and then placed one by one to the nearest available placement site. Abacus [30] is proposed to minimize the squared displacements of the cells optimally. In FastPlace [31], the placeable part of the rows are identified and divided into set of segments. Legalization in FastPlace first targets at moving cells from the segment with too many cells to the neighboring segments iteratively until a target density is met. Cells within one segment are then shifted to satisfy the non-overlapping constraint. Brenner [32] uses a method of partitioning the placement region into bins and then a flow-based approach is used to transport cells among bins such that no bin contains more cells than it can contain. Cells are then locally legalized within a bin.

2.6 Detailed Placement

After global placement, local refinement will be performed to further optimize the objectives while keeping the solution legal. Some common techniques in detailed placement include local reordering in which a small number of neighboring cells are reordered to improve the objectives, local move in which a cell is moved to a nearby position, and local swap in which two closeby cells are swapped. Besides, there are also studies on the problem of Single-row Fixed-order Placement. In this problem, the cells do not change in order and cells in other rows are assumed to be fixed, the problem is to place the cells on the given row to minimize HPWL. Kahng et al. [33] proposed an optimal solution to the problem using dynamic programming. Two similar efficient algorithms are proposed by Refs. [33] and [34] that iteratively place the cells/clusters at their optimal positions and merge the overlapping clusters/cells into bigger clusters, until there is no overlapping. Li et al. [35] proposes a mixed integer programming (MIP) model that can further optimize wirelength and routability of the placement results obtained from several state-of-the-art placers.

3. Routability-driven Placement

Wirelength-driven placement can help to reduce wiring resources and relieve routing congestion to an extent. However, minimizing wirelength excessively can worsen routing congestion because the cells may be placed too close to each other in order to minimize wirelength. An uneven distribution of routing demand will also result in local routing congestion. Wang et al. [36] point out that minimizing congestion is not equivalent to minimizing wirelength. A better way is thus to target at reducing routing congestion directly. However, as explained above, routability also depends on the router to be used. In order to resolve this problem, some efforts were made in the area of modeling routing congestion.

3.1 Congestion Estimation

Routability estimation plays an important role in routability-driven placement. In general, routability in placement refers to the feasibility of generating a successful global routing result. A straightforward way is to call a global router with a placement solution to obtain a routing congestion map. According to the congestion map, the placement is optimized to resolve the routability issues. However, this is impossible because of the long running time of invoking a router and even if running time is not a concern, different routers or the same router but with different settings will generate routing solutions of different congestion.

To estimate routing congestion, an $M \times N$ array of bins called *GCells* covering the routing region is defined. For each boundary edge e between the bins, a term called *routing resource supply* S_e is defined as the maximum number of tracks that can pass through between the two cells. *Routing resource demand* D_e is defined as the number of tracks actually being used. *Routing congestion* occurs at an edge e when the routing demand D_e exceeds the routing supply S_e. When routing congestion occurs at an edge e, the difference between routing demand and supply is called *routing demand overflow* O_e, while *routing congestion* C_e is commonly defined as the ratio of routing demand to supply:

$$O_e = \max(0, D_e - S_e) \tag{14}$$

$$C_e = \frac{D_e}{S_e} \tag{15}$$

Therefore, when routing congestion occurs on an edge e, $O_e > 0$ and $C_e > 1$.

However, the routing resource demand and supply of each edge give a local view of the routing problem. Routaiblity-driven placer or other optimization tools require a more global view of the circuit's routability, or even better, a single metric that can accurately reflect the routing congestion. Some commonly used metrics are evaluated in Ref. [37].

Total overflow (TOF) and **maximal overflow (MOF)** measure the sum and the maximum of routing overflow. When the values are zero, it is estimated that the circuit is routable. However, the metric does not reflect the seriousness of the routing congestion problem. A relatively high overflow may be easily resolved when there are plenty of unused routing resource in the neighborhood region. Beside, the amount of overflow highly depends on the design size, the routing bin (GCell) size, the number of routing layers, etc. The numbers are not intuitive to reflect how serious the routing congestion problem is.

Average net congestion (ACN(x)) is defined as the average congestion of the top $x\%$ congested nets, where the congestion of a net is defined as the maximum congestion among all the global routing edges used by the net. **Worst congestion index WCI(y)** is defined as the number of nets with congestion greater than or equal to $y\%$, where the net congestion is defined in the same way as that in ACN(x). The problem with the two metrics is that they cannot differentiate between a net with a single congested edge and another net spanning multiple congested edges, in which the later one is much harder to resolve.

Average edge congestion ACE(x,y) is proposed by Wei et al. [37]. It is the average congestion of the top $x\%$ congested GCell edges, while ignoring the edges that have more than $y\%$ of the routing tracks blocked. Routing tracks in a GCell can be blocked by routing blockages and interconnection within a GCell.

3.1.1 Probabilistic Model

To estimate routing demand during the placement stage, a common method is to estimate how wires will be placed probabilistically under the current placement and pin distribution. With this information, one can identify the congested locations with information of how much more routing resources are needed probabilistically. A good congestion estimation with probabilistic model should consider the behavior of a global router, such as suppressing usage of via, preference of using simple paths like L-shape and straight line connection, etc.

RISA [38] considers the bounding box of each net and a net weight is computed from the pin count to estimate the routing demand contributed by each net. Reference [39] breaks down multi-pin nets into sets of two-pin nets and uses a probabilistic model to estimate the horizontal and vertical track usage. References [40] and [41] propose an improved probabilistic model with consideration of the router's behaviors of minimizing vias and detours due to blockages, and thus can attain higher accuracy of congestion estimation. RUDY [1] defines *rectangular uniform wire density* d_n of each net n as the ratio of the wire area and the net's bounding box's area. Estimated routing demand at a location (x, y) is the sum of the wire density d_n of all the nets with (x, y) contained in the net's bounding box.

In the recent decade, computational power of the machines running the optimization tools has been improved a lot and efficient global routing algorithms are available. It is generally accepted that actual global routing can be used as a method of routability estimation. Since probabilistic models indirectly model the behaviors of a global router, it is less useful when global routing is fast enough to be called frequently in the optimization tools. Therefore, probabilistic models are less popular in new placement tools.

3.1.2 Global Router Integration

The major purpose of earlier global routers is to guide detailed routing. However, with the increasing demand of routability estimation in placement, recent global routers will also provide a "fast" mode to trade off routing accuracy for running time.

In BonnPlace [10], BonnRouteGlobal [42] is used for congestion estimation. CGRIP [43] uses integer programming (IP) in global routing, which gives high quality global routing solution in a reasonable amount of time. FastRoute [44] features very short runtime by using congestion-driven and via-aware Steiner tree construction followed by layer assignment. It is used in the works of Refs. [45] and [23]. NCTUgr [47] also features fast global routing by using 2D routing followed by layer assignment. The works of Refs. [48] and [46] uses NCTUgr as the routing estimator.

3.1.3 Other Methods

GLARE [37] improves the routing congestion estimation on top of global routing with consideration of local interconnection within a GCell. The local routing demand is estimated in two parts: 1) Rectilinear Steiner trees are constructed to connect the local pins of each local net with two or more pins in the GCell, and the wirelength of the tree segments is evaluated. 2) Local pin density is calculated. The two metrics are used to calculate the amount of routing tracks being blocked by local connections and is added as additional routing demand of the global router.

The work [49] integrates probabilistic modeling and local pin density to estimate routing demand. In NTUPlace [50], routability is optimized by minimizing net congestion in bins. In each bin, a term called *Routable area* is obtained, which is the supply in routing resource in the bin. A weighted overlapping area between a net's bounding box and a bin is taken as the demand in routing resource of the net in the bin. Minimization of the routing congestion is integrated into the objective function, which is solved with a non-linear analytical method.

3.2 Congestion Removal

Once local routing hotspots are identified, we need to resolve the routing congestion. One of the most common ideas is to shift cells in the congested region towards nearby regions with available routing resources. This will result in evening out the routing demands and thus improving the circuit routability. There are several techniques to achieve this routing demand migration.

3.2.1 Cell Inflation

To remove congestion, a congestion map is first obtained by some congestion estimation methods. According to the congestion map, cells in congested regions are identified. These cells are inflated according to how congested the cell location is. Legalization is then applied to the placement solution with the inflated cell sizes. Assuming that the regions with routing congestion usually have high cell density, cells and thus the pins of the cells are spread out from the congested region to the neighboring region with redundant routing resource.

CRISP [51] inflates cells according to the congestion map and the local pin density. The size of a cell c at the i^{th} iteration is calculated as follows:

$$width(c,i) = \max(width(c,i-1)+1, \lceil (1+\alpha \cdot T(c,i)) pin(c) \rceil) \tag{16}$$

where $T(c,i)$ is the number of times the cell c has been in a congested region at the i^{th} iteration, α is a constant called width increment, $pin(c)$ is the number of pins on c and $width(c,i)$ is the width of c at the i^{th} iteration, i.e., $width(c,0)$ is the initial and actual cell size.

SimPLR [52] also considers the amount of routing congestion in the inflation function. Cells in the GCell with serious routing congestion will be inflated more than those in GCell with lower routing demand overflow.

$$\begin{aligned} &width(c,i) = \max(width(c)+1, (1+\theta \cdot T(c,i) \cdot C(g_c)) pin(c)) \\ &C(g_c) = \frac{Demand(g_c)}{Supply(g_c)} \end{aligned} \tag{17}$$

where $T(c,i)$ is the number of times the cell c has been in a congested region, $C(g_c)$ is the congestion of the GCell g_c containing cell c, $Demand(g)$ and $Supply(g)$ are the demand and supply of the routing resources in GCell g, $pin(c)$ is the number of pins on c and $width(c)$ is the actual cell size of c. Unlike CRISP [51], the width increment value, θ (α in CRISP), is dependent on the routing solution, which is calculated as:

$$\begin{aligned} &\theta = \max(0, \gamma \cdot \eta(G) \cdot \xi(G) + \beta) \\ &\eta(G) = \sum_{g \in G} \frac{Demand(g)}{Supply(g)} \\ &\xi(G) = \frac{\sum_{g \in G} \max(0, Demand(g) - Supply(g))}{\sum_{g \in G} Supply(g)} \end{aligned} \tag{18}$$

where G is the set of all GCell, $\eta(G)$ is the total congestion of all GCells, $\xi(G)$ is obtained by dividing the total amount of routing overflow by the total amount of routing supply, which reflects the routing difficulty of the design.

Ripple 2.0 [23] uses a path-based cell inflation method. It searches for the cells actually causing the congestion by tracing the cells connecting to the nets that contribute to the GCell edge's routing demand overflow. In their approach, cell inflation is directional. To resolve routing congestion due to horizontal wires, cells that are causing the congestion are inflated vertically, i.e., the cell heights are increased. Similarly, vertical wire congestion is resolved by inflating cell widths. The inflation function is as follows:

$$\begin{aligned} &width(c) = \max(width(c), \max_{e \in E_c^V}(C(e)) \cdot width(c)) \\ &height(c) = \max(height(c), \max_{e \in E_c^H}(C(e)) \cdot height(c)) \\ &C(e) = \frac{Demand(e)}{Supply(e)} \end{aligned} \tag{19}$$

where E_c^H and E_c^V are the sets of GCell edges corresponding to horizontal wires and vertical wires respectively connecting to cell c, $C(e)$ is the congestion of the GCell edge e, $Demand(e)$ and $Supply(e)$ are the demand and supply in routing resources of the GCell edge e respectively. With such inflation strategy, cells' spreading direction can be controlled and it is found to be more effective in resolving congestion.

BonnPlace [10], [49] inflates cells in a GCell according to the congestion of its incident global routing edge and the local pin density. For each cell c, an inflation ratio $b(c)$ is defined for determining the cell size:

$$s'(c) = s(c) \cdot (1 + b(c)) \tag{20}$$

where $s'(c)$ is the inflated cell size, and $s(c)$ is the actual cell size. Initially, $b(c)$ for each cell is set to be proportional to its pin-to-area ratio. The values are then scaled such that the total inflated area equals $\frac{\tau}{4}$ of the total cell area, i.e.:

$$\begin{aligned} &b(c) \propto \frac{pin(c)}{s(c)} \\ &s.t. \sum_{c \in C} b(c) \cdot s(c) = \frac{\tau}{4} \sum_{c \in C} s(c) \end{aligned} \tag{21}$$

where $pin(c)$ is the number of pins on cell c and τ is a constant. During placement, $b(c)$ is updated according to the estimated congestion of the GCell g in which cell c is located, and the pin density:

$$\begin{aligned} &b(c) = \left(\sum_{1 \le i \le 4} \min(1, 2(C(e_i) - 1)) \cdot \frac{\tau}{5} \right) + \min(1, \alpha P(g)) \cdot \frac{\tau}{5} \\ &C(e) = \max\left(1, \frac{Demand(e)}{Supply(e)}\right) \end{aligned} \tag{22}$$

where e_1, e_2, e_3, e_4 are the four boundaries of GCell g and $C(e)$ is the congestion of e, and $P(g)$ is the pin density of GCell g when it is greater than a threshold or zero otherwise. Therefore, cell is inflated by at most τ.

After inflation, the cells must be spread out in order to even out the routing demand for congestion removal. The rough legalization techniques discussed in Section 2.3.2 can be used. However, there are some rough legalization techniques specially designed for resolving routing congestion. POLAR 2.0 [45] proposes a routability-driven rough legalization, which extends its previous work of Ref. [53] with consideration of routing congestion. In the original work, placement density hotspots are identified as local regions with cell area density exceeding the target density. For each hotspot, a minimal expansion window containing the hotspot is found, which has the total white space area large enough to place all the cells inside the window with the target density. Cells inside the window are then spread evenly within the window to satisfy the density constraint. To consider routability, routing hotspots are identified and the expansion window for resolving a routing hotspot should have enough horizontal and vertical routing resource supply higher than the total routing demand within the window, i.e.,

$$\begin{aligned} &\sum_{i \in W} Supply_H(i) \ge \alpha \times \sum_{i \in W} Demand_H(i) \\ &\sum_{j \in W} Supply_V(i) \ge \alpha \times \sum_{j \in W} Demand_V(i) \end{aligned} \tag{23}$$

where W is the set of all GCell boundaries within the expansion window, $Supply_H(i)$ and $Supply_V(i)$ are the horizontal and vertical routing resource supply of GCell boundary i, respectively, $Demand_H(i)$ and $Demand_V(i)$ are the horizontal and vertical routing demand of GCell boundary i, respectively.

3.2.2 Other Methods

He et al. [46] proposed a post-processing method on global placement to improve circuit's routability. The cells that connect to the wires contributing to routing congestion are identified as problematic cells. For each problematic cell, both the cell and the wires connecting the cell are ripped up from the placement. The cell is relocated to a position without congestion and with the ripped-up wires re-connected. Significant routability improvement is reported.

Ropt [54] shares a similar idea of relocating cells to improve global routing result. For each cell needed to be relocated, the cell and wires connecting the cell are ripped up from the placement. A set of candidate bins around the original position of the cells are evaluated with a cost function considering bin density and the resulting routing congestion. The bin with the lowest cost becomes the target position of the cell.

Cong et al. [55] suggested that the existence of narrow placement channels between fixed macros contributes to routing congestion. The underlaying reason is that cells placed in the narrow regions are difficult to have their nets escaping from the region during routing. The method of blocking narrow channels is implemented by inflating fixed macros according to their distance to neighbouring fixed macros. In this way, the inflated part of the macros covers the narrow channels and no cells can be placed there.

4. Detailed Routability-driven Placement

The ultimate goal of placement is to place a circuit that can finally be routed successfully with timing closure. Therefore, we should definitely consider detailed routability. In detailed routing, exact geometries of where to place the wires will be considered and we thus need to take into account details like pin access, power/ground placement, and design rules like minimum spacing. In today's manufacturing technology with small feature sizes, complex and fine-grain design rules are given for laying out features on the metal layers. The design rules are also much more restrictive due to the many constraints in fabrication and manufacturing. Detailed routing must place the metal wires and vias satisfying the design rules as otherwise, placement may need to be done all over again.

Although routability-driven placemnt has been a hot topic for many years, most of the works handle routing feasibility at the level of global routing. Design rules however will not be considered in global routing. A placement which is routable in the global routing stage may actually be not routable in the detailed routing stage. Works [37], [56], [57] have shown that the predictions by global router on the actual violations encountered in the final detailed routing phase are usually not accurate enough. That means a placement optimized perfectly towards global routing congestion may still be unroutable in detailed routing. To address this issue, we need a much more accurate routability model that can reflect the detailed routability and at the same time, need to develop a placer that can address the potential problems and violations in detailed routing directly.

Exploration in this direction of detailed routing-driven placement is also hindered by the missing of proprietary design data in the publicly available benchmarks. Recently, placement test suites that include fictitious geometry reproducing the essential challenges of real fabrication geometry were released [5], [6]. The availability of these benchmarks will bring quicker discovery of faster and better placement algorithms optimizing towards the real routability.

4.1 Design Rules

In this section, some of the design rules that can potentially be handled directly during the placement stage are introduced.

4.1.1 Non-default Routing (NDR) Net

Some of the nets, especially the net with high fan-out, require larger wire spacing in order to reduce resistive effect. They are usually large networks like the clock net. We have to consider the special design rule for these nets as they will consume more routing resource than the others. Non-default routing rules may specify (1) increased wire spacing for a net, (2) increased wire width for a net or (3) increased via number at selected junctions.

4.1.2 Pin Access

Pin access is the feasibility of a pin to be connected to the metal wires connecting to other pins in the same net. It is a design rule violation when a pin is blocked from accessing by a via or a wire. This happens when a metal-1 pin is under a metal-2 power stripe or when a metal-2 pin overlaps with a metal-2 power stripe. In both cases, the pins are unaccessible and the situations should be avoided. Besides, two cells with pins close to the boundary and connected to non-default routing (NDR) nets cannot be placed too close to each other. Otherwise, one of the pins will be unaccessible.

4.1.3 General Design Rules

The *minimum wire spacing* rule requires a minimum spacing between any two metal edges. The minimum spacing is dependent on the widths of the two adjacent metal objects and the length of the parallel portions between the two adjacent objects. Besides, the *end-of-line* rule imposes a minimum spacing between a wire and a neighboring end-of-line. The minimum spacing is dependent of the position of a third object placed in parallel with the end-of-line object.

4.1.4 Power Network Blockage

Power network is usually evenly distributed over the whole placement region, since every cell being placed have to be powered. It can consume more than 30% of the routing resources on the upper routing layers [5], [6]. They are also wider than signal nets. Early consideration of power net can enhance routability of the placement solution. In general, we can consider a dense Power/Ground (PG) mesh with each routing layer having a uniformly spaced PG rails running in parallel with its preferred routing direction. Rail thickness is constant on each layer but varies between layers.

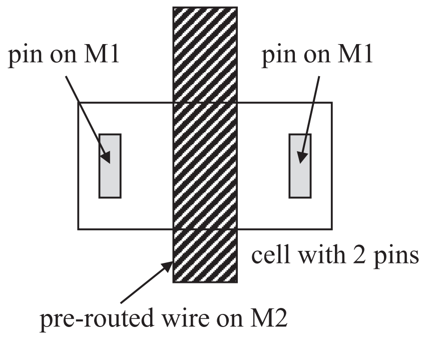

Fig. 4 Valid placement of cells.

4.2 Handling of Design Rule Violations

The works [15], [58], [59] handle pin shorts and pin access problems in the legalization process. M1 and M2 pre-routed wires including power network are considered as placement blockages. As the result, pins will not be placed under pre-routed wires and the pin access problem is thus avoided, although some valid cell placement as shown in **Fig. 4** will also be forbidden.

The work [59] extends Tetris legalizer [29] to consider the minimum spacing requirement. Kennings et al. [58] extended and modified the Abacus legalizer [30]. In Abacus, when a cell is being inserted to one of the rows, the cost is the resulting total displacement of the cells in the row. In the modified Abacus algorithm in [58], the cost becomes a weighted sum of total displacement, total number of design-rule violations, and the cell area overflow of the row as follows:

$$Cost_{c\rightarrow r} = \sum_{c_i \in C_r} DISP_{c_i} + \gamma DRC_{c_i} + \epsilon VIOL_r \tag{24}$$

where $Cost_{c\rightarrow r}$ is the cost of inserting cell c to row r, $DISP_{c_i}$ is the displacement of cell c_i, and DRC_{c_i} is the number of design rule violations. When the total area of the cells assigned to row r exceed the row's area, $VIOL_r$ is the amount of cell area overflow, otherwise it is zero. A row balancing step is used to resolve the cell area overflow problem through moving cells to the neighboring rows. For the remaining DRC violations in each row, an additional cell shifting procedure extending the graph based approach in Ref. [60] is proposed. To resolve DRC violations, the cost of assigning cell c to a placement site s is changed to minimize a weighted sum of the displacement of the cell and the number of DRC violations as follow:

$$Cost_{c\rightarrow s} = DISP_{c,s} + \alpha DRC_{c,s} \tag{25}$$

where $Cost_{c\rightarrow s}$ is the cost of placing cell c to site s, $DISP_{c,s}$ and $DRC_{c,s}$ are the displacement and the number of DRC violations when cell c is placed at the site s.

Wang et al. [48] handles detailed-routability in detailed placement. Their method handles the design rules related to the spacing requirement between cells. To handle cell spacing requirement, a two-step method is proposed. First, cells with spacing violations are inflated and spread to satisfy the violated requirement, provided that such operation produce a legal placement result. Those cells which still have spacing violations are then flipped to see if the number of violations can be reduced.

Kennings et al. [58] also handle the detailed-routability issues in placement. Two techniques of cell moves and cell swaps [34] were used to perturb the placement. Each cell movement is evaluated by a cost with consideration of the total wirelength and design rule violation as follows:

$$Cost = HPWL + w \times DRC \tag{26}$$

where $HPWL$ is the total wirelength, DRC is the number of design rule violations, w is a weight computed as $\beta \times \frac{HPWL}{|N|}$, where β is a constant, $|N|$ is the total number of nets, and $\frac{HPWL}{|N|}$ is thus the average net wirelength.

5. Concluding Remarks

Placement is an important step in the VLSI design cycle that has significant impact on the quality of the final circuit design. As technology node scales down and circuit sizes increases, new concerns and constraints arise that placers today need to handle and resolve. Traditional wirelength-driven placement is the basic on which issues on routability and detailed routability are to be addressed. There are a lot of works and progresses on these placement and routability problems in recent few decades, and there are still more to come. Some of the important open problems are still yet to be studied, for example:

(1) The works [15], [48], [58], [59] introduced in Section 4 handle the design rules related directly to the cell placement, such as pin access and cell spacing, while the design rules related to the spacings of detailed routing wires are ignored.

(2) As mentioned in Section 4, global routability does not reflect the feasibility of detailed routing. Although there are many works on global routability estimation, the detailed-routability estimation is an important issue which is yet to be explored.

Besides the interconnect issues, there are also other significant aspects like timing, power, manufacturing related issues and scalability in placement that need to be addressed.

References

[1] Spindler, P. and Johannes, F.M.: Fast and Accurate Routing Demand Estimation for Efficient Routability-driven Placement, *Proc. DATE* (2007).

[2] Viswanathan, N., Alpert, C.J., Sze, C., Li, Z., Nam, G.-J. and Roy, J.A.: The ISPD-2011 Routability-Driven Placement Contest and Benchmark Suite, *Proc. ISPD* (2011).

[3] Viswanathan, N., Alpert, C., Sze, C., Li, Z. and Wei, Y.: The DAC 2012 Routability-Driven Placement Contest and Benchmark Suite, *Proc. DAC* (2012).

[4] Viswanathan, N., Alpert, C., Sze, C., Li, Z. and Wei, Y.: ICCAD-2012 CAD Contest in Design Hierarchy Aware Routability-Driven Placement and Benchmark Suite, *Proc. ICCAD* (2012).

[5] Yutsis, V., Bustany, I.S., Chinnery, D. and Shinnerl, J.R.: ISPD 2014 Benchmarks with Sub-45 nm Technology Rules for Detailed-Routing-Driven Placement, *Proc. ISPD* (2014).

[6] Bustany, I.S., Chinnery, D., Shinnerl, J.R. and Yutsis, V.: ISPD 2015 Benchmarks with Fence Regions and Routing Blockages for Detailed-Routing-Driven Placement, *Proc. ISPD* (2015).

[7] Alpert, C.J. and Kahng, A.B.: Recent Directions in Netlist Partitioning: A Survey, *Integration*, Vol.19, No.1 (1995).

[8] Karypis, G., Aggarwal, R., Kumar, V. and Shekhar, S.: Multilevel Hypergraph Partitioning: Applications in VLSI Domain, *TVLSI*, Vol.7, No.1 (1999).

[9] Alpert, C., Kahng, A., Nam, G.-J., Reda, S. and Villarrubia, P.: A Semi-Persistent Clustering Technique for VLSI Circuit Placement, *Proc. ISPD* (2005).

[10] Brenner, U., Hermann, A., Hoppmann, N. and Ochsendorf, P.: BonnPlace: A Self-Stabilizing Placement Framework, *Proc. ISPD* (2015).

[11] Alpert, C.J., Kahng, A.B. and Yao, S.-Z.: Spectral Partitioning with Multiple Eigenvectors, *Discrete Applied Mathematics*, Vol.90, No.3 (1999).

[12] Lin, T., Chu, C. and Wu, G.: POLAR 3.0: An Ultrafast Global Placement Engine, *Proc. ICCAD* (2015).

[13] Kim, M.C., Lee, D.-J. and Markov, I.L.: SimPL: An Effective Placement Algorithm, *TCAD*, Vol.31, No.1 (2012).

[14] He, X., Huang, T., Xiao, L., Tian, H. and Young, E.F.Y.: Ripple: A Robust and Effective Routability-Driven Placer, *TCAD*, Vol.32, No.10 (2013).

[15] Darav, N.K., Kennings, A., Westwick, D. and Behjat, L.: High Performance Global Placement and Legalization Accounting for Fence Regions, *Proc. ICCAD* (2015).

[16] Lin, T., Chu, C., Shinnerl, J.R., Bustany, I. and Nedelchev, I.: POLAR: A High Performance Mixed-Size Wirelength-Driven Placer With Density Constraints, *TCAD*, Vol.34, No.3 (2015).

[17] Xiu, Z., Ma, J.D., Fowler, S.M. and Rutenbar, R.A.: Large-Scale Placement by Grid-Warping, *Proc. DAC* (2004).

[18] Brenner, U. and Struzyna, M.: Faster and Better Global Placement by a New Transportation Algorithm, *Proc. DAC* (2005).

[19] Spindler, P., Schlichtmann, U. and Johannes, F.M.: Kraftwerk2 — A Fast Force-Directed Quadratic Placement Approach Using an Accurate Net Model, *TCAD*, Vol.27, No.8 (2008).

[20] Vygen, J.: Algorithms for Large-Scale Flat Placement, *Proc. DAC* (1997).

[21] Eisenmann, H. and Johannes, F.M.: Generic Global Placement and Floorplanning, *Proc. DAC* (1998).

[22] Viswanathan, N. and Chu, C.: FastPlace: Efficient Analytical Placement Using Cell Shifting, Iterative Local Refinement, and a Hybrid Net Model, *TCAD*, Vol.24, No.5 (2005).

[23] He, X., Huang, T., Chow, W.-K., Kuang, J., Lam, K.-C., Cai, W. and Young, E.F.Y.: Ripple 2.0: High Quality Routability-Driven Placement via Global Router Integration, *Proc. DAC* (2013).

[24] Naylor, W.C., Donelly, R. and Sha, L.: Non-linear Optimization System and Method for Wire Length and Delay Optimization for an Automatic Electric Circuit Placer, *US Patent*, 6301693 (2001).

[25] Chen, T.-C., Cho, M., Pan, D.Z. and Chang, Y.-W.: NTUplace3: An Analytical Placer for Large-Scale Mixed-Size Designs With Preplaced Blocks and Density Constraints, *TCAD*, Vol.27, No.7 (2008).

[26] Hsu, M.-K., Chang, Y.-W. and Balabanov, V.: TSV-Aware Analytical Placement for 3D IC Designs, *Proc. DAC* (2011).

[27] Ray, B.N.B. and Balachandran, S.: An Efficient Wirelength Model for Analytical Placement, *Proc. DATE* (2013).

[28] Kahng, A.B. and Wang, Q.: Implementation and Extensibility of an Analytic Placer, *TCAD*, Vol.24, No.5 (2005).

[29] Hill, D: Method and System for High Speed Detailed Placement of Cells within an Itegrated Circuit Design, *US Patent*, 6370673 (2002).

[30] Spindler, P., Schlichtmann, U. and Johannes, F.M.: Abacus: Fast Legalization of Standard Cell Circuits with Minimal Movement, *Proc. ISPD* (2008).

[31] Viswanathan, N., Pan, M. and Chu, C.: FastPlace 3.0: A Fast Multilevel Quadratic Placement Algorithm with Placement Congestion Control, *Proc. ASPDAC* (2007).

[32] Brenner, U.: VLSI Legalization with Minimum Perturbation by Iterative Augmentation, *Proc. DATE* (2012).

[33] Kahng, A.B., Tucker, P. and Zelikovsky, A.: Optimization of Linear Placements for Wirelength Minimization with Free Sites, *Proc. ASPDAC* (1999).

[34] Pan, M., Viswanathan, N. and Chu, C.: An Efficient and Effective Detailed Placement Algorithm, *Proc. ICCAD* (2005).

[35] Li, S. and Koh, C.-K.: MIP-based Detailed Placer for Mixed-size Circuits, *Proc. ISPD* (2014).

[36] Wang, M., Yang, X. and Sarrafzadeh, M.: Congestion Minimization During Placement, *TCAD*, Vol.19, No.10 (2000).

[37] Wei, Y., Sze, C., Viswanathan, N., Li, Z., Alpert, C.J., Reddy, L., Huber, A.D., Tellez, G.E., Keller, D. and Sapatnekar, S.S.: GLARE: Global and Local Wiring Aware Routability Evaluation, *Proc. DAC* (2012).

[38] Cheng, C.-L.E.: RISA: Accurate and Efficient Placement Routability Modeling, *Proc. ICCAD* (1994).

[39] Lou, J., Thakur, S., Krishnamoorthy, S. and Shen, H.S.: Estimating Routing Congestion Using Probabilistic Analysis, *TCAD*, Vol.21, No.1 (2002).

[40] Kahng, A.B. and Xu, X.: Accurate Pseudo-Constructive Wirelength and Congestion Estimation, *Proc. SLIP* (2003).

[41] Saeedi, M., Zamani, M.S. and Jahanian, A.: Prediction and Reduction of Routing Congestion, *Proc. ISPD* (2006).

[42] Müller, D., Radke, K. and Vygen, J.: Faster Min-Max Resource Sharing in Theory and Practice, *Mathematical Programming Computation*, Vol.3, No.1 (2011).

[43] Wu, T.-H., Davoodi, A. and Linderoth, J.T.: GRIP: Global Routing via Integer Programming, *TCAD*, Vol.30, No.1 (2011).

[44] Xu, Y., Zhang, Y. and Chu, C.: FastRoute 4.0: Global Router with Efficient Via Minimization, *Proc. ASPDAC* (2009).

[45] Lin, T. and Chu, C.: POLAR 2.0: An Effective Routability-Driven Placer, *Proc. DAC* (2014).

[46] He, X., Chow, W.-K. and Young, E.F.Y.: SRP: Simultaneous Routing and Placement for Congestion Refinement, *Proc. ISPD* (2013).

[47] Liu, W.-H., Kao, W.-C., Li, Y.-L. and Chao, K.-Y.: Multi-Threaded Collision-Aware Global Routing with Bound-Length Maze Routing, *Proc. DAC* (2010).

[48] Wang, C.-K., Huang, C.-C., Liu, S.S.-Y. and Chin, C.-Y.: Closing the Gap between Global and Detailed Placement: Techniques for Improving Routability, *Proc. ISPD* (2015).

[49] Brenner, U. and Rohe, A.: An Effective Congestion Driven Placement Framework, *Proc. ISPD* (2002).

[50] Hsu, M.-K., Chen, Y.-F., Huang, C.-C., Chen T.-C. and Chang, Y.-W.: Routability-Driven Placement for Hierarchical Mixed-Size Circuit Designs, *Proc. DAC* (2013).

[51] Roy, J.A., Viswanathan, N., Nam, G.-J., Alpert, C.J. and Markov, I.L.: CRISP: Congestion Reduction by Iterated Spreading during Placement, *Proc. ICCAD* (2009).

[52] Kim, M.-C., Lee, D.-J. and Markov, I.L.: SimPL: An Effective Placement Algorithm, *Proc. ICCAD* (2010).

[53] Lin, T., Chu, C., Shinnerl, J.R., Bustany, I. and Nedelchev, I.: POLAR: Placement based on Novel Rough Legalization and Refinement, *Proc. ICCAD* (2013).

[54] Liu, W.-H., Koh, C.-K. and Li, Y.-L.: Optimization of Placement Solutions for Routability, *Proc. DAC* (2013).

[55] Cong, J., Luo, G., Tsota, K. and Xiao, B.: Optimizing Routability in Large-Scale Mixed-Size Placement, *Proc. ASPDAC* (2013).

[56] Alpert, C.J., Li, Z., Moffitt, M.D., Nam, G.-J., Roy, J.A. and Tellez, G.: What Makes a Design Difficult to Route, *Proc. ISPD* (2010).

[57] Liu, W.-H., Koh, C.-K. and Li, Y.-L.: Case Study for Placement Solutions in ISPD11 and DAC12 Routability-Driven Placement Contests, *Proc. ISPD* (2013).

[58] Kennings, A., Darav, N.K. and Behjat, L.: Detailed Placement Accounting for Technology Constraints, *Proc. VLSISoC* (2014).

[59] Huang, C.-C., Chiou, C.-H., Tseng, K.-H. and Chang, Y.-W.: Detailed-Routing-Driven Analytical Standard-Cell Placement, *Proc. ASPDAC* (2015).

[60] Kahng, A.B., Markov, I.L. and Reda, S.: On Legalization of Row-Based Placements, *Proc. GLSVLSI* (2004).

Wing-Kai Chow received his B.Sc. degree from The Hong Kong Polytechnic University (HKPU), Hong Kong, in 2009, and the M.Sc. degree in computer science from The Chinese University of Hong Kong (CUHK), Hong Kong, in 2012, where he is currently pursuing the Ph.D. degree. He was a Research Assistant with HKPU in 2010, and CUHK, from 2010 to 2012. He was a Software Engineering Intern with Cadence Design Systems, USA, in 2014 and 2015. His current research interests are design automation of VLSI, including placement and routing.

Evangeline F.Y. Young received her B.Sc. and M.Phil. degrees in computer science from The Chinese University of Hong Kong (CUHK), Hong Kong. She received the Ph.D. degree from The University of Texas at Austin, USA, in 1999. She is currently a Processor with the Department of Computer Science and Engineering, CUHK. Her current research interests include algorithms and CAD of VLSI circuits. She is now working actively on floorplanning, placement, routing, DFM and algorithmic designs. Prof. Young served on the organization committees of ISPD, ARC, and FPT and the program committees of several major conferences including DAC, ICCAD, ISPD, ASP-DAC, DATE, and GLSVLSI. She also served on the editorial boards of the IEEE TCAD, ACM TODAES and Integration, the VLSI Journal.

(Invited by Editor-in-Chief: *Nozomu Togawa*)