Pin Accessibility-driven Placement Optimization with Accurate and Comprehensive Prediction Model

Suwan Kim and Taewhan Kim
Department of Electrical and Computer Engineering
Seoul National University
Email: suwankim@snucad.snu.ac.kr, tkim@snucad.snu.ac.kr

***Abstract*—The significantly increased density of pins of standard cells and the reduced number of routing tracks at sub-10nm nodes have made the pin access problem in detailed routing very difficult. To alleviate this pin accessibility problem in detailed routing, recent works have proposed to make a small perturbation of cell shifting, cell flipping, and adjacent cells swapping in the detailed placement stage. Here, an essential element for the success of pin accessibility aware detailed placement is the installed cost function, which should be sufficiently accurate in predicting the degree of routing difficulty in accessing pins. In this work, we propose a new model of cost function that is comprehensively devised to overcome the limitations of the prior ones. Precisely, unlike the conventional cost functions, our proposed cost function model is based on the empirical routing data in order to fully reflect the potential outcomes of detailed routing. Through experiments with benchmark circuits, it is shown that using our proposed cost function in detailed placement is able to reduce the routing errors by 44% on average while using the existing cost functions reduce the routing errors on average by at most 15%.**

***Index Terms*—Pin accessibility, placement, standard cell, physical design, prediction model.**

I. Introduction

The pin accessibility problem is one of the main issues in advanced technology nodes [1]–[9]. Although advanced technology nodes have made a significant improvement in chip performance, the complex design rules and lack of routing resources have been accompanied as well. Conventionally, by scaling down the metal pitch and size in direct proportion to the degree of shrinkage in the new technology nodes, the imposed design rules had been achievable. However, due to the different scaling speeds of FEOL (front-end-of-line) and BEOL (back-end-of-line) layers, there is nothing for it but to consider using off-tracks in in-cell routing and the new design rules (e.g., metal-overhang on via) have made the routing problem much more difficult. Furthermore, as the height of standard cells decreases, ensuring the pin accessibility in routing becomes very challenging due to the reduced number of routing tracks and the increased pin density in cells.

The prior approaches to enhancing pin accessibility in detailed placement can be classified as: *algorithmic* and *data-driven*. All existing algorithmic approaches commonly perform two steps: (1) devising a cost function, based on the detailed placement information, to be used for predicting the degree of pin accessibility; (2) performing local perturbations on cell placement through cell shifting, cell flipping, cell swapping with an adjacent cell, and cell replacement. The effectiveness of their approaches relies on how accurately their cost functions predict the pin accessibility.

Noticeable and representative cost functions proposed in the literature are those in [1]–[3]. Taghavi *et al.* [1] developed a cost function $K(C)$ for cell C utilizing features related to pin accessibiltiy: the number of pins in C, the total pin area in C, and the collective bounding box size of all pin pairs in C. However, the cost formulation in [1] explains only the pin configuration on each cell and does not include the net topology information at all, which is one of the critical factors for quantifying the pin accessibility for nets. Seo *et al.* [2] developed a cost function $RPA(p)$, called *remaining pin access points on pin* p, which is a measure indicating how many access points on p remain after accessing its neighboring pins. Even though $RPA(p)$ is a fine-grained formulation in comparison with $K(C)$, it also excludes the net topology information. On the other hand, Ding, Chu, and Mak [3] made one step forward to the cost formulation by taking the net topology information into account in their cost function PAP_{AB}, called *pin access penalty*, which reflects the impact of obstacles in the bounding box enclosed by pin patterns A and B of the same net on the pin accessibility to connect A and B. Although PAP_{AB} incorporates the possible scenarios of accessing points in A and B, it focuses on a pair of pin instances in the net. In other words, for multi-terminal nets, there could be more than one terminal (i.e., pins) or Steiner points, one of which pin A should connect to, but PAP_{AB} completely ignores the consideration of potential access direction and location, ending up exploring all the components exhaustively in the net with pin A.

Recently, a number of data-driven approaches have been proposed to predict the routability of the placement. Chan *et al.* [4] trained models with a number of features (e.g., max. pin density, max. number of edges) from the information of diverse design implementations by using Multivariate Adaptive Regression Splines (MARS) and Support Vector Machine (SVM). Then, they used the set of estimated values of parameters as modeling parameters of a binary classifier developed using SVM algorithm to predict whether a specific placement was routable. Chan *et al.* [5] also used SVM to build a classifier that predicts whether a design rule violation (DRV) has occurred in a given local window. The works in [6]–[8] modeled routability predictors by using pin pattern images as input to build convolutional neural network (CNN) models. Contrary to the role of algorithmic approaches, which

978-3-9819263-6-1/DATE22/©2022 EDAA

is to evaluate the cell/pin whether pin accessibility is very unlikely, the role of data-driven approaches is to predict whether a local tile or window in a chip will contain at least one routing failure. In the view of placement refinement in which local perturbations of the cells are performed, the aforementioned models have the following limitations. The input features of [4], [5] cannot capture the difference of local perturbations (such as cell flipping), if the cells are totally enclosed in a given local window. [8] uses a model with an enormous number of parameters, which is not suitable as a reference model to the placement refinement performing local perturbations repeatedly. Although the works in [6] and [7] can distinguish small perturbations, they require a considerable amount of time for training.

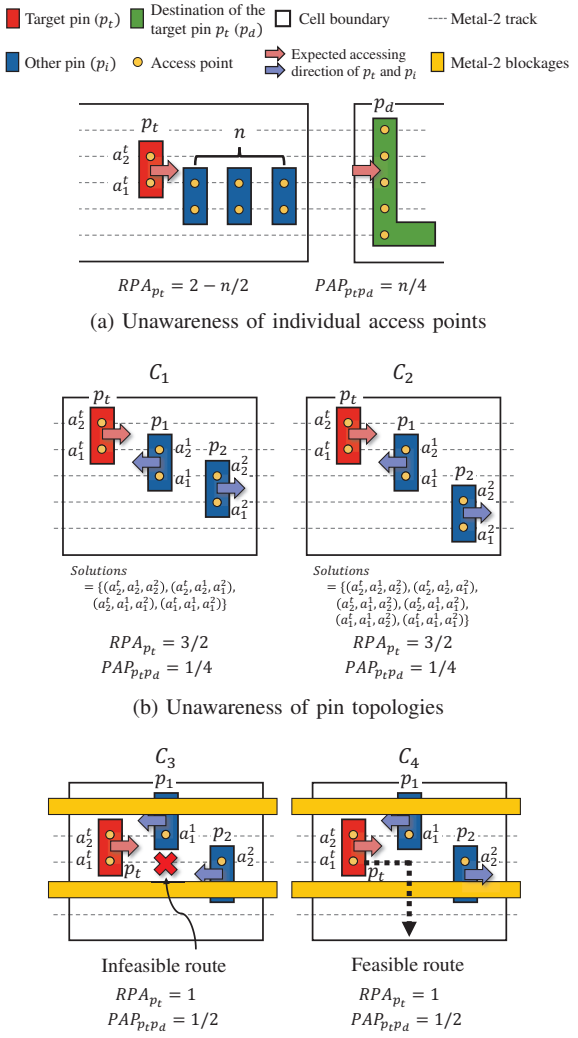
This work belongs to the algorithmic approaches utilizing a cost function consisting of historical routing information, which can be easily combined with the placement refinement. Also, this work aims at overcoming two critical limitations of the previous pin accessibility cost functions: (1) *lack of incorporating pin access direction and distance* and (2) *inaccuracy of incorporating diverse routing scenarios*. Precisely, the main contributions of this work are as follows.

- *Overcoming limitation 1*: We show that identifying the most likely routing direction for accessing a target pin is an essential element for accurate prediction on pin accessibility. By utilizing FLUTE [10] for generating rectilinear minimum Steiner tree and edge shifting [11] for considering routing congestion, we propose a method of *determining a location, from which a direct vertical or horizontal connection to a target pin will most likely occur* in actual routing process.
- *Overcoming limitation 2*: We propose *a comprehensive pin accessibility function for a target pin, which varies in terms of the distance from an accessing location to the pin as well as the obstacles in between them.* Based on the routing data available in existing design implementations, we derive a cumulative conditional distribution function from the routing length statistics. Then, we devise an access distance aware probabilistic prediction function by exploiting the cumulative distribution function.
- *Comparing with existing cost functions*: We *logically and experimentally validate the effectiveness of our cost function* in placement optimization (cell shifting, cell flipping, cell swapping with adjacent cell) by overcoming the inability or lack of the previous cost functions.

II. Observations

We have made a number of prominent observations on how the prior estimation models RPA [2] and PAP [3] lack accuracy in predicting pin accessibility.

- *Lack of incorporating individual access points*: Fig. 1(a) shows a cell placement where target pin p_t with two access points a_1^t and a_2^t is to be routed to the right and the route from a_1^t will be seriously hindered by the n neighbor pins in the cell (n = 3). In such case, according to the cost formulations of *remaining pin access points* (RPA) [2] and *pin access penalty* (PAP) [3], $RPA(p_t) = 2 - n/2$ and $PAP_{p_tp_d} = n/4$. Note



(a) Unawareness of individual access points

(b) Unawareness of pin topologies

(c) Unawareness of access distances

Fig. 1: Limitations of the prior prediction models RPA [2] and PAP [3] of pin accessibility.

that for a target pin in a cell, the lower the value of RPA is or the higher the value of PAP is, the lower the pin accessibility is. Consequently, the cost values indicate that for a large value of n, accessing p_t will be almost impossible. However, p_t has an access point a_2^t of high pin accessibility, which is almost constant, regardless of the value of n. For standard cells in sub-10nm nodes, it is observed that complex gates (e.g., AOI22X1, OAI23X2) with many inputs usually form the serial pin arrangement pattern like the one shown in Fig. 1(a) and such pin pattern is one of the major causes of DRVs.

- *Lack of distinguishing pin topologies*: Fig. 1(b) shows two cell placement instances C_1 and C_2 with three pins each. The configurations of C_1 and C_2 are exactly the same except for the locations of p_2 in C_1 and C_2. We can easily check that C_1 entails four feasible access point combinations (We

count the cases utilizing the metal-2 layer.) for accessing all pins in C_1: (a_2^t, a_2^1, a_2^2), (a_2^t, a_2^1, a_1^2), (a_2^t, a_1^1, a_1^2), and (a_1^t, a_1^1, a_1^2) while C_2 entails six feasible access point combinations: (a_2^t, a_2^1, a_2^2), (a_2^t, a_2^1, a_1^2), (a_2^t, a_1^1, a_2^2), (a_2^t, a_1^1, a_1^2), (a_1^t, a_1^1, a_2^2), and (a_1^t, a_1^1, a_1^2). Note that, the adjacent access points (e.g., (a_1^1, a_2^2) in C_1) of the different pins cannot be chosen together for routing due to the design rule that the via spacing should be larger than the metal pitch. However, each of RPA and PAP fails in clarifying the difference of the degree of pin accessibilities in C_1 and C_2 in that the RPA (PAP) formulation for p_t in C_1 is identical to the RPA (PAP) formulation for p_t in C_2. Moreover, the two formulations of IOC in [2] ($CPAP$ in [3]) evaluating the *cell accessibility* for C_1 and C_2 are the same. Thus, in the light of the prior prediction models, the pin accessibility for C_1 is indistinguishable from that for C_2.

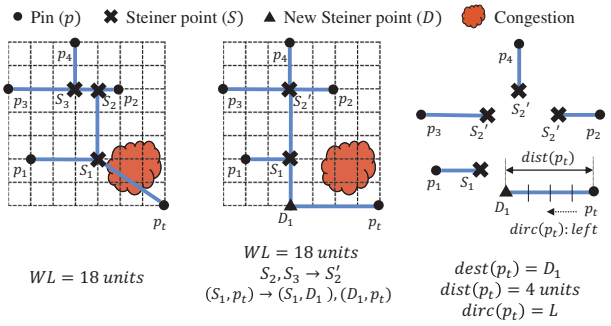
• *Lack of incorporating access distance*: Fig. 1(c) shows two cell placement instances C_3 and C_4 with three pins each. The only difference between the configurations of C_3 and C_4 is the access directions of p_2 in C_3 and C_4. As shown by the dotted black arrow in the configuration of C_4, there can be an alternate route path in a short distance to the right from the access point a_1^t in p_t before conflicting a_2^2 in p_2. On the contrary, C_3 does not entail any feasible route for p_t at all. Nevertheless, the pin accessibility costs RPA [2] and PAP [3], as well as the cell accessibility costs IOC [2] and $CPAP$ [3] each evaluates the same quantities of accessibility for the two configurations, which clearly shows that the prior prediction models are lack of incorporating access distance in their cost formulations.

III. Pin Accessibility driven Placement Optimization

We propose a new methodology of post-placement optimization for enhancing pin accessibility. Our proposed approach based on an **s**tatistical **p**in **a**ccessibility, called SPA, performs the following three steps: (Step 1) *determining most likely access directions and distances* by exploiting the potential rectilinear minimum Steiner trees for nets; (Step 2) *formulating a cost function* that overcomes the limitations of the prior prediction models by utilizing the outcomes obtained from Step 1 and statistical routing data extracted from existing design implementations; (Step 3) *applying cell placement perturbations* with the prediction model built in Step 2.

A. Determination of Potential Access Directions and Distances

For a particular pin, p_t, in a given cell placement, we want to find the location, from which p_t will be most likely accessed at actual routing by using a single vertical or horizontal metal segment. To this end, it is pre-required to estimate the pin-to-pin connection topology for the net of p_t. Since typical routing algorithms generally consider two factors to make a net routing, which are *minimizing wirelength* and *avoiding congestion*, we derive a wirelength driven as well as congestion driven



(a) Wirelength driven routing topology by FLUTE [10]

(b) Congestion driven rectilinear Steiner tree by *Edge-shifting* [11]

(c) Determining accessing point for p_t

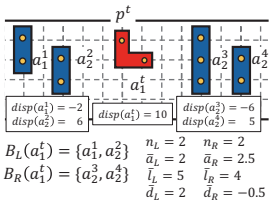
Fig. 2: Stepwise illustration for estimating access distance and direction for target pin p_t.

rectilinear Steiner tree topology by performing the following three steps:

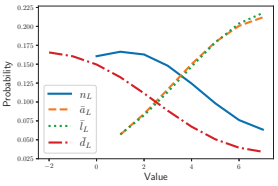
• **Step 1.1** (*Generating wirelength driven routing topology*) In this work, we adopt FLUTE [10], which is a fast lookup table based wirelength estimation technique for constructing the minimum Steiner tree for each net. For example, Fig. 2(a) shows a routing topology for a net with terminals p_t, $p_1, \cdots, p_4$, produced by FLUTE.

• **Step 1.2** (*Generating congestion driven routing topology*) This step is to adjust the edges in the routing topology obtained in Step 1.1, so that the topology becomes a rectilinear Steiner tree topology while taking into account the congestion as well as preserving the wirelength. We adopt *Edge-shifting* technique in [11], which is used to transform a routing topology into a good rectilinear Steiner tree topology based on a fast congestion driven global routing with negligible runtime. For example, Fig. 2(b) shows the Steiner tree topology produced by applying the technique of *Edge-shifting* to the topology in Fig. 2(a). It is seen that by shifting edge (S_2, S_1) in Fig. 2(a) to the left, two Steiner points S_2 and S_3 are merged into one as labeled S_2' shown in Fig. 2(b), and by transforming edge (S_1, p_t) in Fig. 2(b) located on a congestion region into an L-shape edge of the same Manhattan distance to deviate the congestion, edge (S_1, p_t) is split into two edges (S_1, D_1) and (D_1, p_t), generating a new Steiner point D_1 as shown in Fig. 2(b).

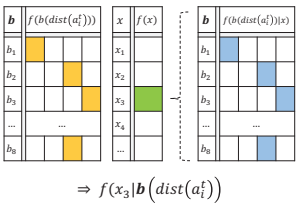
• **Step 1.3** (*Determining the access location and direction*) From the rectilinear Steiner tree, RST, produced in Step 1.2, the direct access location for target pin p_t is the terminal or Steiner point to which p_t is connected by an edge in RST. Let $dest(p_t)$ denote such terminal or Steiner point in RST for p_t. Clearly, the access direction to p_t can be derived from the direction of edge $(p_t, dest(p_t))$. For example, Fig. 2(c) shows the enumeration of all edges in the Steiner tree topology in Fig. 2(b), from which $dest(p_t) = D_1$, $dist(p_t) = 4$ units, and $dirc(p) = \text{L}$ where $dist(p_t)$ indicates the length of edge $(p_t, dest(p_t))$ in RST and $dirc(p) = \text{L}$ or R if $(p_t, dest(p_t))$ is a horizontal edge and $dest(p_t)$ is located on the left or right of p_t, respectively while $dirc(p) = \text{U}$ or B if $(p_t, dest(p_t))$ is a vertical edge and $dest(p_t)$ is located above or below from



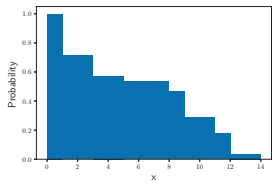
(a) Extracting, from placement data, $\bar{n}_L, \bar{n}_R, \bar{a}_L, \bar{a}_R, \bar{l}_L, \bar{l}_R, \bar{d}_L$, and $\bar{d}_R$ for every access point with $disp(a_i^t)$ by performing Step 2.1.



(b) Deriving a probabilistic function of blockages $f(\boldsymbol{b}(disp(a_i^t)))$ in Step 2.2 to estimate $f(x|\boldsymbol{b}(disp(a_i^t)))$, the probability of connecting a_i^t to a location in *exactly* x *distance* away.



(c) Deriving $f(x|\boldsymbol{b}(disp(a_i^t)))$ utilizing lookup tables consisting of $f(\boldsymbol{b}(disp(a_i^t)))$, $f(x)$, and $f(\boldsymbol{b}(disp(a_i^t))|x)$.



(d) Deriving a conditional cumulative density function $\mathcal{F}(x|\boldsymbol{b}(disp(a_i^t)))$ in Step 2.3 to estimate the probability of connecting a_i^t to a location in *at least* x *distance* away.

Fig. 3: Stepwise procedure for generating a statistical pin accessibility prediction function for SPA.

p_t, respectively.

It is confirmed that the accuracy of estimating the direction of $dest(p_t)$ by our proposed three-step method in Step 1 is 75%.

B. Formulation of Statistical Pin Accessibility

Our pin accessibility model is based on the statistical pin access data taken from the actual routing implementations. Let $\mathcal{F}(x|\boldsymbol{b}(disp(a_i^t)))$ be a conditional cumulative density function that tells the probability of connecting access point a_i^t to a location in *at least* x *distance away* where $disp(a_i^t)$ is set to $1 * dist(a_i^t)$ if $dirc(a_i^t) = R$ while $disp(a_i^t)$ is set to $-1 * dist(a_i^t)$ if $dirc(a_i^t) = L$, and $\boldsymbol{b}(disp(a_i^t))$ is a vector representing the blockages on accessing a_i^t. Thus, the higher the value of $\mathcal{F}\{x = disp(a_i^t)|\boldsymbol{b}(disp(a_i^t))\}$ is, the easier the pin accessibility is. Then, we define our pin accessibility penalty function, called SPA, for a pin p_t as:

$$\mathsf{SPA}(p_t) = \sum_{a_i^t \in A(p_t)} \left(1 - \mathcal{F}(x = disp(a_i^t)|\boldsymbol{b}(disp(a_i^t)))\right) \quad (1)$$

in which $A(p_t)$ is the set of all access points in p_t. The term in *Eq.1* indicates a statistical estimation of the unlikelihood of accessing p_i^t in line with $disp(\cdot)$ of its access points. Then, our pin access penalty for a cell c_k is defined as:

$$\mathsf{CSPA}(c_k) = \sum_{p_t \in P(c_k)} \mathsf{SPA}(p_t) \quad (2)$$

where $P(c_k)$ is the set of pins in cell c_k.

Fig. 3 shows a stepwise procedure for the derivation of $\mathcal{F}(x|\boldsymbol{b}(disp(a_i^t)))$, which is composed of four steps:

• **Step 2.1** We extract, from cell placement information, access direction $dirc(a_i^t)$ and access distance $dist(a_i^t)$ for every access point a_i^t by performing Step 1. The two extracted parameters are combined into $disp(a_i^t)$. Then, we collect all access points located on the same horizontal access track as that of a_i^t and then split them into two sets $B_L(a_i^t)$ and $B_R(a_i^t)$, one containing access points on the left side of a_i^t and the other on the right side. For every access point with the same value of $disp(a_i^t)$, we define the blockage vector for a_i^t as:

$$\boldsymbol{b}(disp(a_i^t)) = [n_L, n_R, \bar{a}_L, \bar{a}_R, \bar{l}_L, \bar{l}_R, \bar{d}_L, \bar{d}_R]. \quad (3)$$

where $n_{(\cdot)}$ is the number of pins in set $B_{(\cdot)}(a_i^t)$, $\bar{a}_{(\cdot)}$ is the average number of access points for pins containing access points in set $B_{(\cdot)}(a_i^t)$, $\bar{l}_{(\cdot)}$ is the average distance between a_i^t and an access point in $B_{(\cdot)}(a_i^t)$, and $\bar{d}_{(\cdot)}$ is the average $disp(\cdot)$ value for all access points in $B_{(\cdot)}(a_i^t)$. Fig. 3(a) shows an example of the extraction of vector features, in which for $disp(a_i^t)$ = 10, its $B_L(a_i^t)$ and $B_R(a_i^t)$ are built, from which a vector of $[n_L, n_R, \bar{a}_L, \bar{a}_R, \bar{l}_L, \bar{l}_R, \bar{d}_L, \bar{d}_R]$ is derived.

• **Step 2.2** We generate, from routing data, a conditional probability density function, $f(x|\boldsymbol{b}(disp(a_i^t)))$, to measure the probability of connecting a_i^t to the location in *exactly* x *distance away* given the blockages $\boldsymbol{b}(disp(a_i^t))$. Then, assuming all the features are mutually independent to each other, we apply Bayes theorem to define f as:

$$f(x|\boldsymbol{b}(disp(a_i^t))) = \frac{f(x)}{f(\boldsymbol{b}(disp(a_i^t)))} \cdot \prod_{b_i \in \boldsymbol{b}(disp(a_i^t))} f(b_i|x) \quad (4)$$

It should be noted that since the amount of sampled features for every exact value of $\boldsymbol{b}(disp(a_i^t))$ is not sufficient in practice, we discretize the $\boldsymbol{b}(disp(a_i^t))$ values and collect samples for each of the discretized values. Fig. 3(b) shows the probabilistic distribution of $f(b_i|x)$ for $b_i \in B_L(a_i^t)$ with $disp(a_i^t) = 5$. If the number of pins n_L increases, the probailistic value $f(b_i|x)$ decreases. The same tendency for $\bar{d}_L$ can be observed, nevertheless, the opposite trends are shown for $\bar{a}_L$ and $\bar{l}_L$. To predict $\mathcal{F}(x|\boldsymbol{b}(disp(a_i^t)))$, we propose a lookup table(LUT)-based approach as shown in Fig. 3(c). The values of $f(x)$, $f(\boldsymbol{b}(disp(a_i^t)))$, and $f(\boldsymbol{b}(disp(a_i^t))|x)$ are stored for every discretized value of x and b_i. The elements in LUTs are called for calculating *Eq.4* followed by performing the cumulation to build *Eq.5*.

• **Step 2.3** We derive a conditional cumulative density function $\mathcal{F}(x|\boldsymbol{b}(disp(a_i^t)))$ for $f(x|\boldsymbol{b}(disp(a_i^t)))$, which measures the probability of connecting a_i^t to a location in *at least* x *distance*:

$$\mathcal{F}(x|\boldsymbol{b}(disp(a_i^t))) = f(x' > x|\boldsymbol{b}(disp(a_i^t))) \quad (5)$$

Note that $\mathcal{F}$ is calculated in a reverse order of x. This function tells the degree of easiness of connecting a_i^t in line of $\boldsymbol{b}(disp(a_i^t))$. Fig. 3(c) illustrates $\mathcal{F}(x|\boldsymbol{b}(disp(a_i^t)))$ for $disp(a_i^t) = 8$ and $\boldsymbol{b}(disp(a_i^t)) = [2, 2, 4, 4, 4, 2, 4, 4]$.

C. Cell Placement Optimization

The ultimate objective of performing cell flipping, cell swapping, and cell shifting is to minimize the quantity of $\mathsf{TSPA}(n)$:

$$\mathsf{TSPA}(n) = \sum_{i=1}^{n} \mathsf{CSPA}(C_i) \tag{6}$$

where n is the total number of cells in the placement. Our proposed cell placement optimization consists of two steps:

• **Step 3.1** We employ the same procedure of the first phase used in [3]. That is, we perform cell flipping and adjacent cell swapping row-by-row by applying dynamic programming formulation.

• **Step 3.2** We make a slight modification of the second phase used in [3]: instead of using a linear programming (LP) for cell shifting as does in [3], in which the shifting positions are represented as continuous values, i.e., the placement optimality is not assured, we formulate the placement optimization by cell shifting into a dynamic programming in a way to place the cells on placement sites, ensuring placement optimality. Our proposed recursive formula for Step 3.2 can be formally expressed as:

$$\begin{aligned}\mathsf{TSPA}(i+1)^{j}_{d_{i+1}} = min_{d_i \in [-M,M]} \{ & \mathsf{CSPA}(C_i(d_i)) \\ & + \Delta\mathsf{CSPA}(C_i(d_i), C_{i+1}(d_{i+1})) \}\end{aligned} \tag{7}$$

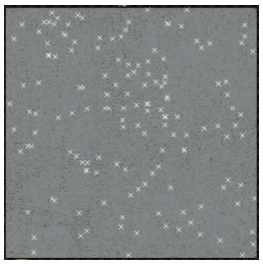
where $\mathsf{TSPA}(i)^{j}_{d_i}$ is the total cost for the row j with cells C_1, C_2, ..., C_i, ..., C_{n^j} such that C_i has displacement of d_i, $\mathsf{CSPA}(C_i(d_i))$ indicates a cost for C_i with displacement of d_i, and $\Delta\mathsf{CSPA}((C_i(d_i), C_{i+1}(d_{i+1})))$ indicates the amount of cost increment when C_{i+1} with displacement of d_{i+1} abuts to C_i with d_i. Note that $d_i \in [-M, M]$ and $d_i \in \mathbb{Z}$, so cells should be placed respecting the placement sites.

IV. Experimental Results

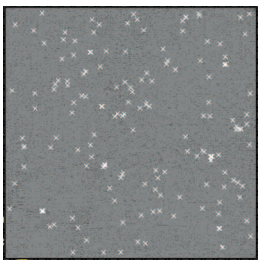
We implemented Step 1 to Step 3 using C++ programming language. In our experiments, we used ASAP 7nm cell library and design rules in [12] as our process design kit (PDK), and OpenCores [13] were used for benchmark circuits. We synthesized the benchmark circuits by using Synopsys Design Compiler [14] and performed P&R by using Cadence Innovus [15]. We conducted experiment with an Intel-i7 CPU at 4.7GHz, 64GB memory. To construct LUTs, more than 8 millions routing data are extracted from various designs.

A. Assessing Effectiveness of Prediction Models

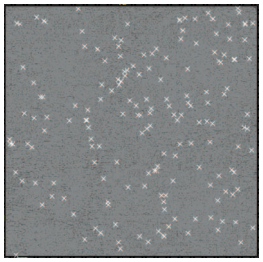
Table I shows the comparison of the implementations in terms of the number of metal-2 short violations (#DRVs), the total wirelength (WL), the average amount of cell instance displacement (Dis.) and the runtime (Rt.) in seconds used by the placement optimizations employing the conventional pin accessibility prediction models of RPA [2] and PAP [3], and employing our prediction model SPA in terms of routing tracks. The second, third, fourth, and fifth columns in Table I list the total number of cell instances (#Cells), total number of nets (#Nets), the number of metal-2 short violations (#DRVs), and total wirelength (WL) in μm of the initial cell placements of the corresponding benchmark circuits, respectively. For a fair comparison, we apply the same placement optimization techniques (cell flipping, swapping, shifting) for RPA, PAP, and our SPA, and set metal-2 shorts as a major design rule violation caused by pin inaccessibility.



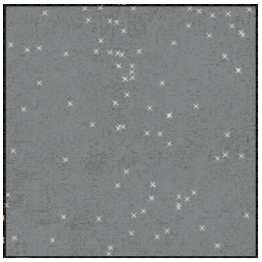
(a) Final implementation of initial placement (Innovus): 169 DRVs



(b) Final implementation of optimized placement using RPA [2]: 166 DRVs



(c) Final implementation of optimized placement using PAP [3]: 178 DRVs



(d) Final implementation of optimized placement using our SPA: 94 DRVs

Fig. 4: Distribution of DRVs in the final implementations of circuit JPEG produced by using initial placement (Cadence Innovus), placement optimization with pin accessibility prediction models RPA, PAP, and our SPA.

As shown in Table I, the amount of improvements by the placement optimization using RPA [2] and PAP [3] fluctuates severely, much less DRVs for some circuits (e.g. AES_128) while more DRVs for some other circuits (e.g., VGA_ENH and ECG). However, the placement optimization using our prediction model SPA shows consistent improvements for all benchmark circuits. In short, the placement optimization using PAP [3] rarely resolves DRVs, even using a considerable amount of cell displacement i.e., 7.47 tracks on average whereas the placement optimization using RPA [2] produces 14.88% less DRVs at the expense of using total cell displacement of 4.83 tracks. On the other hand, the placement optimization using SPA produces 44.64% less DRVs, even using the cell displacement of just 1.09 tracks on average.

Fig. 4 shows the distribution of DRVs in the final implementations of circuit JPEG produced by using initial placement (Cadence Innovus), placement optimization with pin accessibility prediction models RPA, PAP, and our SPA. The examples clearly show that our model SPA is very accurate to predict the pin accessibility in routing.

B. Assessing Placement Optimization Techniques using SPA

We applied three options of placement optimization techniques using prediction model SPA: (option 1) cell flipping and adjacent cell swapping, (option 2) cell shifting only, (option 3) cell flipping, swapping, and shifting. Table II summarizes the final implementations produced by applying the

TABLE I: Comparison of the final (P&R) implementations in terms of the number of metal-2 short violations (DRVs), the total wirelength (WL), the average amount of cell displacement (Dis.), and the runtime (Rt.) in seconds used by the placement optimizations employing the conventional prediction models of RPA [2] and PAP [3], and employing our model SPA.

	# Insts.	# Nets	#DRVs	WL	RPA [2]				PAP [3]				SPA			
					#DRVs	WL	Dis.	Rt. (s)	#DRVs	WL	Dis.	Rt. (s)	#DRVs	WL	Dis.	Rt. (s)
wb_dma	2639	2856	2	29416	0	28789	5.39	23	3	28938	7.79	30	0	29055	1.16	50
ac97	7094	7178	8	85363	7	85876	7.01	73	5	85278	8.87	193	3	84962	1.18	148
wb_conmax	18751	19881	111	399416	94	400792	3.59	251	84	403885	6.10	1273	52	401169	1.26	441
eth	34833	34961	17	819228	17	813742	1.22	655	12	813956	12.06	1688	5	814175	1.18	996
vga_enh	52863	52952	15	1424293	17	1416555	1.49	940	19	1417456	13.31	2931	7	1416168	1.63	1189
ecg	86417	87348	9	1197246	18	1198536	4.83	1176	14	1195355	7.43	3649	8	1194365	1.05	3770
aes_128	102145	102403	112	1486193	66	1466148	4.00	1186	94	1469997	7.20	3445	71	1472133	1.26	4574
nova	149901	150711	40	2967761	31	2957750	2.78	2431	46	2962665	7.33	5616	29	2952277	1.27	4781
tate_pairing	196094	196872	21	2302767	13	2303958	6.32	1893	22	2302097	6.62	5512	10	2299403	0.83	10272
jpeg	250976	252546	169	2943080	166	2905967	6.47	3710	178	2904573	6.53	16387	94	2881223	0.99	14382
Avg (Impr.)			1	1	**0.8512** (**14.88**%↓)	**0.9944** (**0.56**%↓)	**4.8346**	**1**	**0.9464** (**5.36**%↓)	**0.9948** (**0.52**%↓)	**7.4679**	**3.3004**	**0.5536** (**44.64**%↓)	**0.9920** (**0.80**%↓)	**1.0914**	**3.2906**

TABLE II: Comparison of the final (P&R) implementations produced by applying combinations of optimization techniques using SPA.

	Cell flip. + swap.			Cell shift.			Cell flip. + swap. + shift.		
	#DRV	WL	Dis.	#DRV	WL	Dis.	#DRV	WL	Dis.
wb_dma	2	29091	1927	1	28846	1193	0	29055	3063
ac97	1	85349	5424	2	84593	3079	3	84962	8392
wb_conmax	89	401153	14014	88	400487	10407	52	401169	23567
eth	12	814678	11661	20	814345	30572	5	814175	41033
vga_enh	11	1416682	53249	14	1419987	36429	7	1416168	86010
ecg	17	1194470	33315	4	1195709	59151	8	1194365	90617
aes_128	76	1472868	93871	58	1468017	36816	71	1472133	128354
nova	40	2951195	84117	36	2953158	115877	29	2952277	191034
tate_pairing	16	2291203	82305	24	2297780	84159	10	2299403	163549
jpeg	212	2918072	160168	195	2921843	92985	94	2881223	248514
Avg (Impr.)	**0.9444** (**5.56**%↓)	**0.9941** (**0.59**%↓)	**0.5989**	**0.8770** (**12.30**%↓)	**0.9949** (**0.51**%↓)	**0.5220**	**0.5536** (**44.64**%↓)	**0.9920** (**0.80**%↓)	**1.0914**

three options, which clearly support that our pin accessibility prediction model SPA fits very well to all the placement perturbation techniques.

V. Conclusion

This work addressed the problem of alleviating pin inaccessibility. We claimed that an essential element for the success of pin accessibility aware detailed placement was the installed cost function, which should be not only computationally efficient but also sufficiently accurate in predicting the degree of routing difficulty in accessing I/O pins. In this respect, we proposed a new model of a cost function that was comprehensively devised to overcome the limitations of the prior ones. Precisely, unlike the conventional cost functions, our proposed cost function model was based on the empirical routing data in order to fully reflect the potential outcomes of detailed routing. Through experiments with benchmark circuits, it was shown that using our proposed cost function in detailed placement was able to reduce the routing errors by 44% on average while using the existing cost functions reduced the routing errors by at most 15%.

Acknowledgement: This work was supported in part by Samsung Electronics Company, Ltd. under IO201216-08205-01, FOUNDRY-202010DD003F, and EDA cluster project, in part by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MEST) under 2020R1A4A4079177 and 2021R1A2C2008864, in part by the Institute of Information and communications Technology Planning and Evaluation (IITP) grant funded by Korea government (MSIT) under 2021-0-00754, Software Systems for AI Semiconductor Design), in part by National R&D Program through NRF funded by Ministry of Science and ICT under 2020M3H2A1078119, and in part by the BK21 Four Program of the Education and Research Program for Future ICT Pioneers, Seoul National University in 2021. The EDA tool was supported by the IC Design Education.

References

[1] T. Taghavi, Z. Li, C. Alpert, G.-J. Nam, A. Huber, and S. Ramji, "New placement prediction and mitigation techniques for local routing congestion," 2010.
[2] J. Seo, J. Jung, S. Kim, and Y. Shin, "Pin accessibility-driven cell layout redesign and placement optimization," in *DAC*, 2017.
[3] Y. Ding, C. Chu, and W.-K. Mak, "Pin accessibility-driven detailed placement refinement," in *ISPD*, 2017.
[4] W.-T. J. Chan, Y. Du, A. B. Kahng, S. Nath, and K. Samadi, "Beol stack-aware routability prediction from placement using data mining techniques," in *ICCD*, 2016.
[5] W.-T. J. Chan, P.-H. Ho, A. B. Kahng, and P. Saxena, "Routability optimization for industrial designs at sub-14nm process nodes using machine learning," in *ISPD*, 2017.
[6] T.-C. Yu, S.-Y. Fang, H.-S. Chiu, K.-S. Hu, P. H.-Y. Tai, C. C.-F. Shen, and H. Sheng, "Pin accessibility prediction and optimization with deep learning-based pin pattern recognition," *TCAD*, 2020.
[7] ——, "Lookahead placement optimization with cell library-based pin accessibility prediction via active learning," in *ISPD*, 2020.
[8] R. Liang, H. Xiang, D. Pandey, L. Reddy, S. Ramji, G.-J. Nam, and J. Hu, "Drc hotspot prediction at sub-10nm process nodes using customized convolutional network," in *ISPD*, 2020.
[9] S. Kim, K. Jo, and T. Kim, "Boosting pin accessibility through cell layout topology diversification," in *ASPDAC*, 2021.
[10] C. Chu, "Flute: Fast lookup table based wirelength estimation technique," in *ICCAD*, 2004.
[11] M. Pan and C. Chu, "Fastroute: A step to integrate global routing into placement," in *ICCAD*, 2006.
[12] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy, and G. Yeric, "Asap7: A 7-nm finfet predictive process design kit," *Microelectronics Journal*, 2016.
[13] Oliscience, "Opencores," 1999. [Online]. Available: https://opencores.org
[14] "Synopsys Design Compiler." [Online]. Available: https://www.synopsys.com
[15] "Cadence Innovus." [Online]. Available: https://www.cadence.com