Pseudopin Assignment with Crosstalk Noise Control

Chin-Chih Chang and Jason Cong, *Fellow, IEEE*

***Abstract*—This paper presents a new pseudopin assignment (PPA) algorithm with *crosstalk noise control* in *multilayer gridless general-area routing*. We propose a two-step approach that considers obstacles and minimizes the weighted sum of total wire length and the estimated number of vias under crosstalk noise constraints. We test our algorithms on a set of MCM examples and a set of standard-cell examples. Without crosstalk noise control in PPA, the average noise in the MCM test cases after detailed routing is 0.13–0.22 V_{DD} with up to 11% of nets larger than 0.3 V_{DD}. However, if the noise constraint of each net is set to 0.3 V_{DD} in PPA, the average noise in each case reduces to 0.11–0.15 V_{DD} (15%–31% reduction) with no crosstalk noise violations. Most of the nets in our standard-cell test cases do not have noise problems. Our PPA algorithms still give better noise distributions and have 1%–10% noise reduction on the global nets in these standard-cell test cases. Even without ripup and reroute, the detailed routing completion rate is 93%–99% and the average vias per net is only 0.7–1.4 for our MCM test cases and 1.0–1.7 for our standard-cell test cases.**

***Index Terms*—Deep submicrometer, gridless routing, noise analysis, routing.**

I. Introduction

In a typical hierarchical routing system, a global router determines wirings in a rough scale (in terms of routing regions) and a detailed router determines the exact wirings within each routing region. In order to build a bridge between global routing and detailed routing, we need to determine the wire crossing locations on the region boundaries. A wire crossing point is called a *pseudopin* in this paper. The problem of determining the pseudopin locations is called the PPA (PPA) problem. Because PPA determines the wire ordering and spacing to a large extent, it can be used effectively for wire length minimization, via minimization, and crosstalk noise control. We are interested in the problem of PPA with via minimization, wire length minimization, and crosstalk noise control in hierarchical multilayer gridless general-area routing.

In [18], a two-layer grid-based PPA algorithm is discussed. Their heuristic algorithm optimizes the alignment of pseudopins, but does not consider crosstalk. There are some studies on controlling crosstalk noise in detailed or channel routing (e.g., [2], [3], [14], [15], [17], [20], [25]) or in global routing (e.g., [28], [29]). Although the crosstalk noise estimations during detailed routing can be accurate, the freedom to control crosstalk noise is restricted. On the other hand, although crosstalk noise control in global routing may have more flexibility, the estimations can not be very accurate without detailed considerations on wire ordering, spacing, and the complications from obstacles and gridless layouts. In [26], crosstalk noise is considered in a pseudopin[1] assignment step. Their algorithm inserts pseudopins on each boundary one by one with a priority ordering and then performs a space relaxation algorithm to further separate pseudopins. Their greedy algorithm may lack a global view to align the pseudopins of the same nets.

In this paper, we propose a new PPA algorithm to control the crosstalk noise and minimize a weighted sum of the number of vias and wire length in multilayer gridless general area routing. Our algorithm takes the obstacles into consideration by decomposing the tile boundaries into intervals and then solves the PPA problem in two steps: coarse pseudopin assignment (CPPA) and detailed pseudopin assignment (DPPA). In CPPA, each pseudopin is estimated with a crosstalk-safe spacing from its noise constraint and assigned to an interval. Our CPPA algorithms are efficient graph routing algorithms that minimize the weighted sum of wire length and vias. They also ensure that every interval has enough space for all the pseudopins assigned to it. In DPPA, each pseudopin is assigned to an exact location and crosstalk noise constraints must be satisfied. Our DPPA algorithm determines pseudopin ordering and then aligns pseudopins of the same net under crosstalk constraints.

II. Problem Formulation

We are interested in the PPA problem for a multilayer gridless area routing system with obstacles. The inputs of the problem consist of a multilayer global routing solution, a set of design rules, and a set of crosstalk constraints. We assume that the global router uses a reserved layer model, which means each layer has a preferred routing orientation (horizontal or vertical); obstacles are also allowed. We also assume that the global router divides the routing area regularly into an array of rectangular tiles. For each net, the global routing solution determines which *tiles* and *layers* it should go through without giving the exact wire crossing locations. We need to determine the wire crossing locations before a detailed router can route each tile independently. Since these wire crossing locations act just like pins in detailed routing, we shall call them "pseudopins" in this paper. On the other hand, we call the original pins "real pins" to distinguish them from pseudopins.

Assignment of pseudopins may have significant impact on the wire length, the number of vias, and crosstalk noise in the final layout generated by the detailed router. For example, Fig. 1

Manuscript received November 11, 2000. This work was supported in part by DARPA/ETO under Contract DAAL01-96-K-3600 managed by the U.S. Army Research Laboratory and by the Semiconductor Research Corporation under Contract 98-DJ-605. This paper was presented in part at the International Symposium on Physical Design, San Diego, CA, April 2000. This paper was recommended by Associate Editor D. Hill.

The authors are with the Computer Science Department, University of California, Los Angeles, CA 90095 USA (e-mail: cchang@cs.ucla.edu; cong@cs.ucla.edu).

Publisher Item Identifier S 0278-0070(01)03086-X.

[1]In [26], pseudopin is called crosspoint.

shows two PPAs of the same global routing solution on 3×4 tiles. The tile boundaries are shown as dotted lines. Pseudopins are labeled 1–12; real pins are labeled $a1$, $a2$, $b1$, $b2$, $c1$, and $c2$; the grey areas are obstacles. The possible detailed routings according to the PPA are also shown in the figure. The narrow solid lines represent wires on layer 1 and the wide solid lines represent the wires on layer 2. The shaded areas indicate the space between the wires of net $b1$–$b2$ to the wires that are separated by the minimum spacing to them. We can see the total coupled length (length of shaded areas) is roughly 4 (tile widths) in Fig. 1(a), but decreases to 2 (tile widths) in Fig. 1(b). The detour on net $b1 - b2$ is roughly 1 (tile height) in Fig. 1(a) and 1.5 (tile height) in Fig. 1(b). This example shows different PPA solutions can lead to considerably different via counts, wire lengths, and capacitance coupling among nets.

Our objectives of the PPA are to determine the locations of pseudopins to minimize a weighted sum $\alpha TL + \beta VC$ of the total wire length TL and the estimated number of required vias VC under the crosstalk constraints. We assume the weights α and β are given by the user. We choose this objective because crosstalk noise usually only needs to be controlled in a safe range, but the wire length and via minimization is usually desired. Note that minimizing the estimated number of vias means more alignments on pseudopins and less routing resources used by vias, thus generating more routable problem instances for detailed routing.

We estimate the total wire length by the summation of the Manhattan distances between adjacent pins (real or pseudo) of all nets. We shall explain how the crosstalk noise is estimated in the next section. The details of via estimation are explained in Section III-A.

A. Crosstalk Noise Estimation

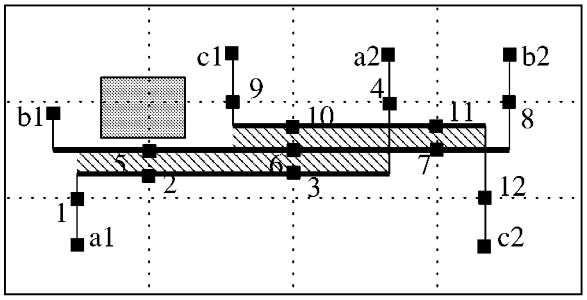
To estimate the crosstalk noise in PPA, we need to estimate the routing of each net and compute the resistance and capacitance from the estimated routing. The routing of each net is estimated by a set of wire segments that correspond to pseudopins. We use seg_p to denote the wire segment corresponds to pseudopin p. If a pseudopin p is on the boundary between tiles T_1 and T_2, the length of the wire segment seg_p is estimated by the center-to-center distance between tiles T_1 and T_2.

Under this assumption, we can estimate the resistance and capacitance for each wire segment. If a pair of pseudopins on the same boundary were assigned adjacent to each other, we could know the coupling length and separating distance between these two wire segments. Therefore, we can estimate the coupling capacitance by a table lookup method [8] (used in our approach). Alternatively, we may also use analytical formulae.

From the above estimated resistance and capacitance together with driver and receiver characteristics information, we can estimate the crosstalk noise in PPA by any crosstalk modeling, including those in [12], [19], [24], [25], and [27].

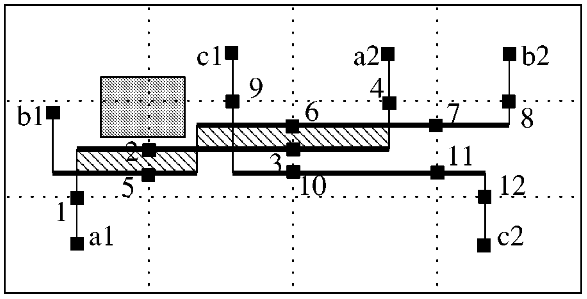
In our implementation, we use a simple closed-form conservative formula for two-terminal nets described in [25] to calculate the crosstalk noise on each wire segment.[2] According to

[2]If there are multiple-terminal nets, we calculate the noise as if all the wire segments of the same net were on a simple path to simplify the noise computation and give a conservative upper-bound estimation on crosstalk noise.



Vias = 6
Coupling capacitance on net b1-b2 ≈ 4
Detour on net b1-b2 ≈ 1

(a)



Vias = 8
Coupling capacitance on net b1-b2 ≈ 2
Detour on net b1-b2 ≈ 1.5

(b)

Fig. 1. Impacts of PPA. (a) PPA with less vias and detours, but larger coupling on net $b1$–$b2$. (b) PPA with smaller coupling on net $b1$–$b2$, but more vias and detours.

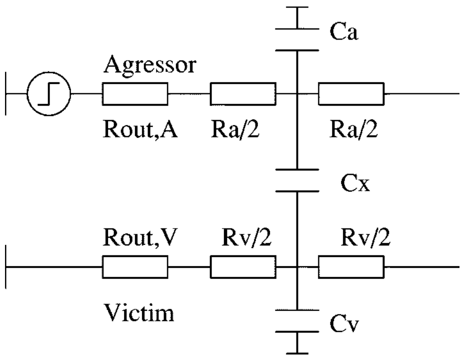


Fig. 2. Crosstalk calculation.

[25], the peak crosstalk noise V_{noise} for the circuits in Fig. 2 can be estimated by the following formula:

$$V_{\text{noise}} = \frac{V_{DD}C_x}{\dfrac{R_{\text{out},A}}{R_{\text{out},V} + R_v/2}C_a + C_v} \qquad (1)$$

where the aggressor is driven by a step voltage source of V_{DD} with intrinsic resistance of $R_{\text{out},A}$ and the victim is connected to ground via its intrinsic resistance $R_{\text{out},V}$. The intrinsic capacitances of the two lines are C_a and C_v, line resistances are R_a and R_v, and the coupling capacitance between the aggressor and victim is C_x.

We choose such a simple formula in our implementation because of the following reasons.

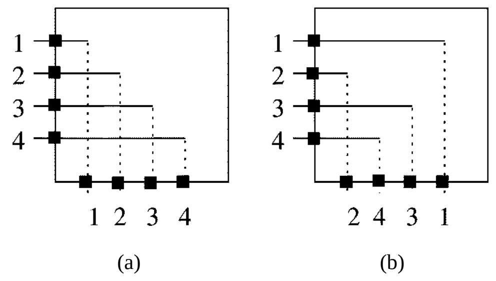


Fig. 3. Net ordering on adjacent layers. (a) Vertical next ordering (1 2 3 4). (b) Vertical next ordering (2 4 3 1).

1) The detailed routing results are still unknown during PPA. Therefore, the noise calculation in PPA will be rough estimations due to the error in estimating coupling capacitance even if the most accurate noise estimation models are used. It is not necessary to use complicated and accurate noise models in PPA. What we need is a conservative model to absorb the estimation errors. According to [25], the simple formula in (1) reports noise always between 10% and 20% higher than AS/X, an IBM circuit simulation tool similar to SPICE.
2) We need a formula that is efficient to evaluate for runtime reasons.

Although we are using the above formula to estimate crosstalk noise, we would like to emphasize that our algorithm is not based on the assumption of which noise model is used. Our algorithm can use any other reasonable noise models such as the 2-π model recently proposed in [10].

We assume the system clock is divided into n user-defined windows (time buckets) [22]. The noise effect in one window will not last to another window. Therefore, we do not need to add up noise on different windows. For each victim net, we only need to add up the noise from its active neighboring aggressor nets within each window.

The benefit of using time buckets is that more aggressive designs are allowed. However, it also requires the knowledge of the logic switching behaviors to determine the value of n and whether there is noise concern between any pair of nets in a window.

Our algorithm simply assumes this information is given by users. If the user does not have such information, our default assumption is that the noise between every net pair should be considered ($n = 1$).

We use a simple assumption that pessimistically estimates the crosstalk noise on a net is the summation of all the crosstalk noise on all the segments of the net. In the case that a more accurate crosstalk modeling is used (e.g., it needs to penalize coupling at the receiver more than coupling at the driver), we can use a weighted sum to calculate the crosstalk with a proper choice of weights.

We use this simple weighted summation method for crosstalk noise for fast noise estimation with the cost of possible overestimation of the crosstalk noise.

B. Layer by Layer Approach

We observe that the assignment of pseudopins in one layer has little affect on PPAs on different layers. For example, Fig. 3

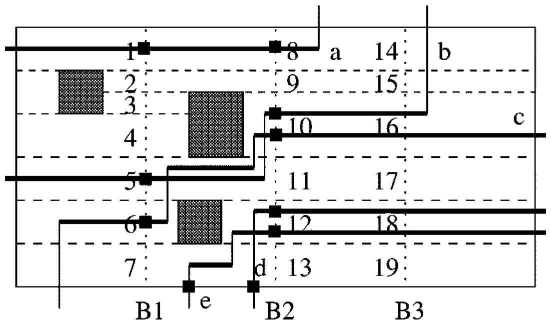


Fig. 4. Tile boundary decomposition and via estimations.

shows that we can permute the pseudopins on the vertical layer without changing the PPA on the horizontal layer. Note that the estimated number of vias and total wire lengths are not changed in these two assignments, although the noise estimations on the horizontal layer will change slightly. However, such change is usually much smaller compared to the change due to ordering or spacing of the pseudopins on the same layer.

Assigning pseudopins one layer at a time can reduce the problem complexity and does not sacrifice too much solution quality. Furthermore, because each pseudopin is confined on a single tile boundary, we do not need to work on the entire layer, assigning pseudopins one row (or column) at a time is good enough. Because the assignment on a row or a column is similar, we will focus on the PPA on a row of tiles in the later discussions.

III. PPA Algorithm

The crosstalk constrained PPA problem is an NP-hard problem even when we only consider a degenerated problem to determine if a feasible PPA exists on a single tile boundary. This can be proved by a simple reduction from the Hamiltonian path problem (a proof similar to that in [15] will be provided in the Appendix).

Our PPA algorithm is a heuristic algorithm that consists of a tile boundary decomposition which partitions tile boundaries into intervals, a CPPA step which assigns pseudopins to intervals, and a DPPA step which assigns the exact locations of pseudopins within each interval. Our algorithm has a similar flavor as the pin assignment algorithm used in [4].

A. Tile Boundary Decomposition

We first decompose boundaries to a set of intervals by maximum horizontal strips. The *maximum horizontal strips*, which were first defined in [23], are strips (rectangles) that form a partition on the empty space in a routing region such that no strip is horizontally adjacent to any other strip. In our algorithm, the rectangular objects are obstacles, real pins, or the projections of real pins from adjacent layers.[3] Fig. 4 shows an example of the maximum strips formed on a row of four tiles. The grey areas are rectangular objects, which are obstacles or real pins. The dashed

[3]In fact, each rectangle is expanded by half of the minimum spacing on that layer. Our algorithm can further cut strips that are too tall (compared to an input parameter) to make sure that we have enough partitions on the routing areas to avoid trivial coarse assignment.

lines are the horizontal lines shot from the top and bottom of the obstacles. The decomposed intervals are labeled 1–19.

The above tile boundary decomposition allows us to accurately estimate the minimum number of required vias under the reserved layer model without knowing the exact pseudopin locations. We use the following approximation to simplify the estimation: we only consider obstacles on the same layer and assume the space on the adjacent layers is always available for making connections.

If two pseudopins p_1 and p_2 are on the same layer and assigned to intervals of strips s_1 and s_2, we only need to check if s_1 and s_2 horizontally overlap. If p_1 is on a horizontal layer and p_2 is on a vertical layer, we only need to check on if s_1 horizontally overlaps with s_2. Fig. 4 shows several examples of the via estimation patterns on boundaries $B1$ and $B2$. The via estimations on pseudopins between $B1$ and $B2$ are 0, 2, 4, 1, and 3 for nets a, b, c, d, and e, respectively.

B. CPPA

In CPPA, we assign pseudopins to intervals. The noise control in CPPA is through space reservation on each pseudopin. The idea is to reserve more space for pseudopins that are more likely to have noise problems such that the pseudopins with potential noise problems can have more space to separate themselves from other pins in DPPA. Our CPPA algorithm reserves space for each pseudopin by estimating a crosstalk-safe spacing for each pseudopin, which will be discussed later.

After the crosstalk-safe spacing for each pseudopin is calculated, the noise control is implicitly done when we resolve the congestion in assigning pseudopins to intervals. Our CPPA algorithm will simply focus on the objective of minimizing a weighted sum $\alpha TL + \beta VC$ of the estimated total wire length TL and the estimated number of vias VC.

Since we do not have the exact pseudopin locations for wire length calculation, we approximate the location of a pseudopin p by the location of the center of the interval that pin p is assigned to.

A CPPA is feasible if all the intervals have enough space for the pseudopins assigned to them. The CPPA problem is an NP-hard problem that can be proved by a simple reduction from the set partition problem to a CPPA problem on a single boundary (a proof will be provided in the Appendix).

The crosstalk-safe spacing for a pseudopin is estimated by assuming that the pseudopin is adjacent to a pair of pseudopins which have the average capacitance, resistance, and driver/receiver characteristics. We assume each pseudopin has a noise budget that can be calculated from the noise constraints.[4] If a pseudopin has a noise budget B with the estimated total capacitance, resistance, and driver/receiver characteristics, we can calculate the maximum allowed coupling capacitance

$$C_x = \left(\frac{R_{\text{out},A}}{R_{\text{out},V} + R_v/2} C_a + C_v \right) B/V_{DD}$$

[4]In our implementation, we evenly distribute the noise constraint of each net to every pseudopin p in the net according to the length of the wire segment seg_p (defined in Section II-A). However, any clever algorithm which distributes noise budgets can be used.

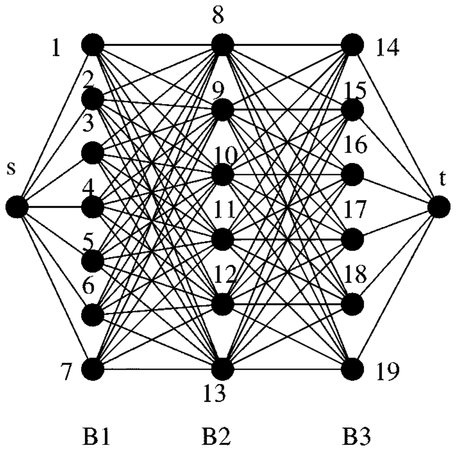


Fig. 5. Coarse routing graph for CPPA.

on this pseudopin from (1). From C_x, we can find the minimum separation distance to its neighbor by interpolation in the capacitance lookup table. This calculated minimum separation distance is our estimated crosstalk-safe spacing for the pseudopin.

1) Coarse Routing Graph: To solve the CPPA problem, we first generate a coarse routing graph $G = (V, E)$ from the boundary decomposition. The vertex set V consists of the intervals and the connection points that are either real pins or pseudopins. The edges in E connect vertex pairs which can reach each other without crossing a tile boundary.[5] For example, Fig. 5 shows the routing graph for a layout in Fig. 4 with two connecting points s and t representing a net that needs to go through the boundaries $B1$, $B2$, and $B3$.

Each edge (u, v) in the routing graph is assigned a cost $d(u, v)$, which is a weighted sum $\alpha h + \beta c$ of the Manhattan distance h between the center of the two intervals and the estimated number of vias c to connect pins on u and v.

It is easy to show that a CPPA for a net connecting from v to u corresponds to a path from v to u in the coarse routing graph and routing cost (weighted sum of wire length and via count) is the sum of the edge costs of the path. Therefore, for a subproblem of the CPPA that assigns a single net, we can use the shortest path algorithm to find the minimum cost assignment.

2) CPPA Algorithms: Based on the optimal assignment of a single net (shortest path algorithm), we implemented two approaches to solve the CPPA problem. The first one is a *net-by-net* approach that just applies the shortest path algorithm to assign nets one by one (with the capacity of each vertex considered). Because it is a straightforward implementation using the single net shortest-path algorithm for CPPA one by one, we will not discuss its details. The second one is an *iterative deletion* approach (similar to the concept first used in global routing [4], [11]), which works one boundary at a time and simultaneously assigns the unassigned nets crossing the boundary with their best assignments (shortest paths).

We first describe the overall flow of our iterative deletion based CPPA algorithm. It works one boundary at a time. At each

[5]In order to make use of the intervals which are too short for assigning any pseudopin, our routing graph has more vertices. For each interval I_v, we introduce two vertices v_{lo} and v_{hi}. If a pseudopin p is assigned to v_{lo} (v_{hi}), it means p is aligned to the bottom (top) of I_v and may cover several short intervals if v is too short for p.

iteration, it checks on the boundaries that are not yet processed and selects a boundary B_x with the most number of pseudopins. For each boundary B_x, it iteratively applies an iterative deletion based multinet CPPA algorithm until all the nets that cross B_x have been assigned. Because the CPPA problem is NP-hard, we do not have an efficient algorithm that can guarantee that we can always find feasible assignments for all nets. If our algorithm can not find feasible assignments, it would do ripup and reroute to enhance the chance of finding a feasible solution.

Before we explain our iterative deletion based multinet CPPA algorithm, we need some definitions to help our algorithm explanations. For a pseudopin p on boundary B_x, we use n_p to denote the subnet in the current row that p belongs to. This subnet connects pins from pin s_p to pin t_p. The coarse routing graph of the subnet n_p contains vertices s_p, t_p, and vertices (corresponding to available intervals) from boundaries B_{sb_p} to B_{tb_p}. The set of the vertices on boundary B_i is denoted as V_i. The shortest path cost from vertex u to vertex v is denoted as $D(u, v)$. We define a vertex set $A(p) = \{v \in V_x \mid D(s_p, v) + D(v, t_p) = D(s_p, t_p)\}$. The vertex set $A(p)$ on boundary B_x is the vertex set that can permit some shortest paths from s_p to t_p. The set $A(p)$ can be found by computing the shortest path from s_p for all vertices in the coarse routing graph and computing the shortest path backward from t_p for vertices between boundary B_x and B_{tb_p}.

The details of our iterative deletion based multinet CPPA algorithm working on a boundary B_x is explained below. For each pseudopin p in V_x, it runs the shortest path algorithm to find $A(p)$ and assign p to all the vertices in $A(p)$. It reports a failure and stops the program if there is no path from s_p to t_p even after the ripup and reroute algorithm has been applied.[6] It also records the shortest path cost of n_p as $C(p)$. Since we have many duplicate assignments, some vertices in V_x can be too crowded. The algorithm then iteratively selects an assignment $\sigma_{p,v}$, which assigns a pseudopin p to a vertex v, and deletes the assignment $\sigma_{p,v}$ until there are no more overly congested vertices. The selection criteria for the assignment $\sigma_{p,v}$ for deletion are: 1) the vertex v is one of the most crowded vertices; 2) pseudopin p has the most number of alternative assignments among the pseudopins that has an assignment in v.

After we have no more overly congested vertices, our algorithm goes through every unassigned pseudopin p on boundary B_x. If pseudopin p has some assignment that has not yet been deleted and the subnet n_p can still be assigned with the cost $C(p)$, we use the shortest path for n_p to assigned all the pseudopins in n_p. Please note that some subnet n_p may not be assigned with shortest path cost $C(p)$ because some vertices become unavailable due to assignments from other nets or all its assignment have been deleted. The pseudopins that are not assigned in the current pass will be assigned in later passes with higher costs.

A summary of the iterative deletion based CPPA algorithm is shown below.

```
1. while there are unassigned boundaries
2.   select the most crowded unassigned
       boundary $B_x$
3.   while there are unassigned pseu-
       dopins in $B_x$
4.     for each pseudopin $p$ not yet as-
         signed on $B_x$
5.       compute $A(p)$ and $C(p)$
6.       assign $p$ to all the vertices in
           $A(p)$
7.     while there are vertices that are
         too congested
8.       delete an assign $\sigma_{p,v}$ that satis-
           fies
           a. vertex $v$ is one of the most
              congested vertices
           b. pseudopin $p$ has the most al-
              ternative assignments in $v$
9.     assign pseudopins in subnet $n_p$ if
         $n_p$ can still be assigned with
         cost$C(p)$
```

3) Speed-Up the CPPA for a Single Net: The CPPA for a single net is the basic building block for both of our net-by-net and iterative deletion CPPA algorithms. We need to apply this algorithm at least once for every net in the net-by-net CPPA algorithm and maybe many times for each net in the iterative deletion CPPA algorithm.

A straightforward implementation of the shortest path algorithm for directed acyclic graph (DAG) has a complexity of $O(V + E)$ time.[7] Since the coarse routing graph is very dense [i.e., $E \approx O(V^2)$], the complexity for the algorithm is about $O(V^2)$, which can be time consuming.

Because the special cost structure of the CPPA problem, we are able to find efficient algorithms (linear time in practice) for the shortest path problems in CPPA. The key idea of our algorithms is to avoid explicitly generating and visiting all the edges while maintaining the optimality of the shortest path algorithm. The reduction in complexity from $O(V^2)$ to almost linear time has great impacts on the runtime our CPPA algorithm. We have seen more than ten times reduction in runtime in our test cases after we switched our implementations to the enhanced version.

If we do not consider the cost of wire length, we shall present later in this section an algorithm to cut down the complexity of the shortest path algorithm to $O(dV)$, where d is the maximum via count per edge. Since d is a small number ($d = 4$ in our formulation), we have a linear time algorithm in practice.

If we consider the weighted cost of wire length and via counts, we shall present later in this section an $O(d\beta V)$ algorithm to calculate the shortest path for the case that all the costs are rounded to some integral multiples of certain unit and the cost for a via is β. When the β is small enough, we still get a linear time algorithm in practice.

Because the exact pseudopin locations are not determined in CPPA, the wire lengths calculated in CPPA are still some

[6]During the ripup and reroute, if we find certain boundary has been ripped-up too many times, we will reduce the crosstalk-safe spacing estimation of some pseudopins and try again with different net ordering. If the boundary has been ripped-up too many times and no more spacing reduction can be done, we will report failure.

[7]For a set S, we will just use S for $|S|$ in the big-O notation when there is no confusion.

rough approximations. We do not need the length calculation to be very precise and we can use larger granularity for the wire length. If we choose a larger granularity for wire length, we can normalize α and β to smaller values such that our algorithm can run faster. For example, we set the cost of one via equal to the cost of a wire of length ten times pitches (minimum wire width and spacing) in our experiments. We also selected ten times pitches as the minimum length unit, i.e., all the lengths are rounded to integral multiples of this unit. In this case, we have $\alpha = 1$ and $\beta = 1$.

We first provide some definitions for our explanations.

Definition 1: Given the source s, the cost for a path $(s, v_1, v_2, \ldots, v_k, v)$ from source s to a vertex v is $d(s, v_1) + d(v_1, v_2) + \cdots + d(v_k, v)$, we define $D(v)$ is the minimum cost of all the paths from s to v.

Definition 2: For a vertex u and a vertex v, we define $DT(u, v) = D(u) + d(u, v)$.

The value of $DT(u, v)$ is the shortest path cost to v under the restriction that edge (u, v) is the last edge in the path.

4) Single Net CPPA for Via Cost Only: We first discuss the case that wire length is not considered ($\alpha = 0$). In this case, the cost $d(u, v)$ between two vertices u, v are the via estimation $via(u, v)$ of the edge. Remembering that a vertex v on boundary i corresponds to an interval I_v, we denote $r(v)$ as the farthest location that we can push I_v horizontally toward boundary $i+1$ without hitting any obstacles. For example, in Fig. 4, we have $r(6) < r(3) = r(4) < r(5)$. In our via estimation, if $r(v) \geq r(u)$, we have $d(v, x) \leq d(u, x)$ for any vertex x on the next boundary except for the case that $d(u, x)$ equals to zero [in this case, $r(u) = r(v)$ and they both reach next boundary $i + 1$].

For a vertex u and a vertex w on the same boundary, all the edges coming out from u will be pruned if the first condition of the following is satisfied; all the edges with costs that are not equal to zero will be pruned if the second condition is satisfied:

1) $D(w) + d < D(u)$, where d is maximum via count per edge;
2) $D(w) \leq D(u)$ and $r(w) \geq r(u)$.

In the first case, *any* edge (u, x) is pruned because a path consists of a shortest path from source to w and (w, x) is always shorter than a path goes through u to x ($DT(w, x) = D(w) + d(w, x) \leq D(w)+d < D(u) \leq D(u)+d(u, x) = DT(u, x)$). In the second case, any edge (u, x) *which cost is not zero* is pruned by a similar reason ($DT(w, x) = D(w) + d(w, x) \leq D(w) + d(u, x) \leq D(u) + d(u, x) = DT(u, x)$). We say u is *dominated* by w or w *dominates* u if any of the above conditions is satisfied.

A simple example to demonstrate the dominating relations among vertices is shown in Fig. 6. It contains several vertices on boundaries i and $i + 1$. For each vertex v, we place it to the center of I_v. For a vertex v in boundary i, we also draw a horizontal stick to show the value of $r(v)$. We assume the maximum via count per edge $d = 4$. Assume the shortest path cost on boundary (stage) i has been calculated and the costs are: $D(a) = 8$, $D(b) = 7$, $D(c) = 8$, $D(d) = 9$, and $D(e) = 12$. We can show that vertex e is dominated by vertex b because $D(b) + 4 < D(e)$. Vertex e is also dominated by vertex d because $D(d) < D(e)$ and $r(d) \geq r(e)$. Vertex a dominates vertex c because $D(a) \leq D(c)$ and $r(a) > r(c)$.

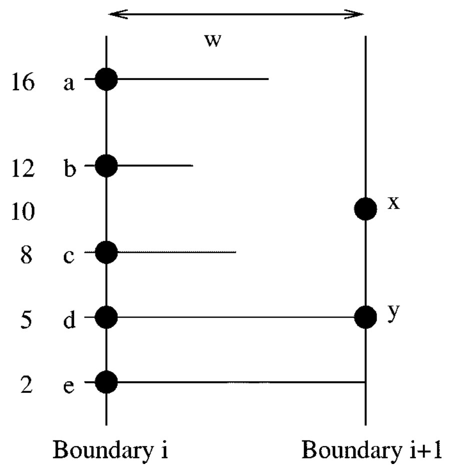


Fig. 6. Vertex dominating.

We call a set DS in stage i is a *dominating set* on stage i if any vertex in stage i is dominated by some vertex in the set DS. For the example in Fig. 6, a set $\{a, b, d\}$ is a dominating set on stage i. A set $\{a, b, c, d\}$ is also a dominating set by definition.

Given a dominating set on stage i, the shortest path cost to a vertex x on stage $i + 1$ can be found by the following formula:

$$D(x) = \min\{DT(y, x) \mid d(y, x) = 0 \text{ or } y \in DS\}.$$

Since any vertex u in stage i is dominated by some vertex v in DS, unless $d(u, x) = 0$, vertex u can be discarded when considering the shortest paths to x because $DT(u, x) \geq DT(v, x)$ by the definition.

The above formula basically says that the shortest path cost to a vertex x can be found by checking on its adjacent edges to the previous stage that have costs equal to zero and the edges from the vertices in a dominating set of the previous stage. Therefore, when we know the shortest path to all the vertices in stage i and a dominating set DS_i, the shortest path cost can be found by checking at most $(|DS_i| + 1) \cdot |V_{i+1}|$ edges, where V_{i+1} is the vertex set on stage $i + 1$.

We shall show next that we can have a dominating set of size at most $d + 1$ in any stage. Given a set SC of vertices in stage i of the same cost, the vertex u that has the largest $r(u)$ will dominate any other vertex in SC. Therefore, we can form a dominating set that has one vertex for each cost in stage i. Let mc be the minimum cost in stage i, we can drop any vertex with cost $c > mc + d$ from the dominating set because they will be dominated by the dominating vertex of cost mc. We now have a dominating set of size at most $d + 1$. In fact, we can further reduce the size of a dominating set if any of the remaining vertices is dominated by another vertex in the dominating set.

Since finding a dominating set only takes $O(dV_i)$ comparisons on stage i and we visit at most only $(d+2)V_i$ edges when we calculate the shortest path cost on stage i, we have an $O(dV)$ time shortest path algorithm.

Theorem 1: If only via cost is considered, CPPA on a single net can be done in $O(dV)$, where d is the maximum via count per edge.

5) Single Net CPPA for Combined Length and Via Costs: We now discuss how we handle the combined length and via cost in our shortest path algorithm. Under this formulation, the edge cost $d(u, v)$ between two vertices u, v is now a weighted sum $\alpha \cdot length(u, v) + \beta \cdot via(u, v)$, where α and β are user specified weights; $length(u, v)$ is the Manhattan distance between the center of interval I_u to the center of interval I_v; $via(u, v)$ is the via cost estimation. Please note $length(u, v) = |y_u - y_v| + tile\ width$, where y_u and y_v are the y-coordinates of the centers of the intervals I_u and I_v.

Because the cost of a path contains the length measurement now, there is no longer a set of vertices that can dominate all the other vertices. The algorithm in the previous subsection can no longer be applied. However, we shall show that we can define two dominating sets for each vertex. Furthermore, a sequence of these dominating sets can be calculated efficiently and we can have an efficient shortest path algorithm.

We first modify the dominating conditions with the aid of a new definition of a modified cost $DL(u, v) = D(u) + \alpha \cdot length(u, v)$. We can rewrite the dominating conditions on vertices w and u for a vertex x in the next stage:

1) $DL(w, x) + \beta d < DL(u, x)$;
2) $DL(w, x) \leq DL(u, x)$ and $r(w) \geq r(u)$.

By similar arguments as in the previous section, an edge (u, x) can be pruned in the shortest path consideration because $DT(w, x) \leq DT(u, x)$ in both cases. We say that w x-dominates u and u is x-dominated by w if any of the above conditions is satisfied. We use the same example in Fig. 6 to demonstrate the new definition. We have labeled the y-coordinates of the vertices in left-hand side of the graph. Assume we have calculated the shortest path costs for all the vertices in stage i: $D(a) = 8$, $D(b) = 7$, $D(c) = 8$, $D(d) = 9$, and $D(e) = 12$. We also assume $\alpha = 1$, $\beta = 1$, and $d = 4$. We can calculate $DL(a, x) = D(a) + 16 - 10 + w = w + 14$, $DL(b, x) = D(b) + 12 - 10 + w = 9 + w$. In this case, vertex a is x-dominated by b because $DL(b, x) + d = w + 13 < DL(a, x)$.

The dominating sets of a vertex x in stage $i+1$ are defined for two subsets of the vertices in stage i: TOP_x and BOTTOM_x. The set TOP_x is formed by the following: we first shoot a horizontal line from the center of interval I_x, any vertex u with the center of I_u that is above or right on the horizontal line is in TOP_x. The rest of the vertices in stage i are in BOTTOM_x. For example, in Fig. 6, vertices a and b are in TOP_x and vertices c, d, and e are in BOTTOM_x.

A set $DST \subseteq \mathrm{TOP}_x$ is a x-top-dominating set if any vertex in TOP_x is x-dominated by a vertex in DST. A set $DSB \subseteq \mathrm{BOTTOM}_x$ is a x-bottom-dominating set if any vertex in BOTTOM_x is x-dominated by a vertex in DSB. Obviously, for any vertex x, we can find an x-top-dominating set and an x-bottom-dominating set with sizes less than $\beta d + 1$ because we can have one x-dominating vertex for each cost and any vertex v with $DL(v, x)$ larger than $DL(u, x) + \beta d$ is x-dominated by u, where u is the vertex with the smallest $DL(u, x)$.

We now discuss how we can find a sequence of top-dominating sets for all vertices on stage $i+1$ in $O(\beta d V_i + V_{i+1})$ time. We first show an efficient way to generate x_2-top-dominating set from a x_1-top-dominating set if x_1 is above x_2 on stage i. In the case that x_1 is above x_2, if $u \in \mathrm{TOP}_{x_1}$, we will have $length(u, x_2) = length(u, x_1) + distance(x_1, x_2)$, where $distance(x_1, x_2)$ is the distance between the centers of intervals I_{x_1} and I_{x_2}. We now have $DL(u, x_2) = DL(u, x_1) + \alpha \cdot distance(x_1, x_2)$. For our example in Fig. 6, $DL(a, y) = D(a) + 16 - 5 + w = D(a) + 16 - 10 + 10 - 5 + w = DL(a, x) + 10 - 5 = DL(a, x) + distance(x, y)$. Similarly, $DL(b, y) = DL(b, x) + distance(x, y)$.

The above formula shows us that if a vertex $u \in \mathrm{TOP}_{x_1}$ is x_1-dominated by a vertex $w \in \mathrm{TOP}_{x_1}$, u is x_2-dominated by w for x_2 (in the example, a is x-dominated by b and a is also y-dominated by b). Therefore, if a set DST_{x_1} is an x_1-dominating set for TOP_{x_1}, it still x_2-dominates all the vertices in TOP_{x_1} for the vertex x_2. We can construct an x_2-top-dominating set by updating $DL(u, x_2) = DL(u, x_1) + \alpha \cdot distance(x_1, x_2)$ for every u in DST_{x_1} in $O(\beta d)$ and compare them against $DL(v, x_2)$ for $v \in \mathrm{TOP}_{x_2} - \mathrm{TOP}_{x_1}$ in $O(\beta d|\mathrm{TOP}_{x_2} - \mathrm{TOP}_{x_1}|)$ time. For the example in Fig. 6, we have an x-top-dominating set $\{b\}$. We calculate $DL(b, y) = DL(a, x) + distance(x, y) = 11 + w + 5$. We then checks on the members of $\mathrm{TOP}_y - \mathrm{TOP}_x$ with $\{b\}$ one by one. For the vertex c, we have $DL(c, y) = 8 + 8 - 5 + w = 11 + w$. Because $r(c) > r(b)$ and $DL(c, y) < DL(b, y)$, vertex b is y-dominated by c. We should update the dominating set to $\{c\}$ by deleting b and adding c. For the vertex d, we have $DL(d, y) = 9 + w$. Therefore, vertex c is y-dominated by d. The final y-top-dominating set is $\{d\}$.

If we do a top-down sweep on the vertices on stage $i+1$, every vertex in stage i appears just once in $\mathrm{TOP}_{x_{j+1}} - \mathrm{TOP}_{x_j}$ for some j. Furthermore, $\mathrm{TOP}_{x_{j+1}} - \mathrm{TOP}_{x_j}$ can also be found efficiently by interleaving a top-down sweep on stage i with the sweep on stage $i+1$. Therefore, we can construct top-dominating sets for all the vertices in stage $i+1$ in $O(\beta d V_i + V_{i+1})$. Similarly, we can find all the bottom-dominating sets for all the vertices in stage $i+1$ in $O(\beta d V_i + V_{i+1})$ by a bottom-up sweep on stage $i+1$. Since $\mathrm{TOP}_x \cup \mathrm{BOTTOM}_x = V_i$, any vertex v in V_i is dominated by some vertex in TOP_x or BOTTOM_x. Therefore, we can find the shortest path to x by just checking on vertices in TOP_x, BOTTOM_x, and the vertex u with $via(u, x) = 0$. As a result, we have an $O(d\beta V)$ time shortest path algorithm.

Theorem 2: If the cost for an edge of h length and v vias is $\alpha h + \beta v$, where α, β, h, and v are integers, CPPA on a single net can be done in $O(\beta d V)$, where d is the maximum number of vias per edge.

C. DPPA

After the CPPA step, every pseudopin is assigned to some interval. The DPPA then assigns pseudopins to the exact locations inside each interval. pseudopins of the same strip (defined by the tile boundary partition) are assigned together in a way somewhat like channel routing. Each subnet n is first assigned as a single wire segment w_n, i.e., all the pseudopins in subnet n are aligned on a straight line. We first determine the ordering and spacing of these wire segments. As in channel routing, in which the number of tracks required may exceed the channel density, we may have assignments that exceed the strip height if we insist that all the pseudopins in each subnet b must be aligned. If this happens, we will apply an alignment algorithm to break up the alignments and introduce jogs to resolve the problem. Instead of demanding all pseudopins of the same net must be aligned, the alignment algorithm will just try to align as many pseudopins as possible.

The details of the ordering and spacing algorithm and the alignment algorithm are discussed in the next two sections.

1) Ordering and Spacing Algorithm: Our ordering and spacing algorithm works on one strip at a time. It is a simple iterative packing algorithm that assumes pseudopins of a subnet n are aligned as one single wire segment w_n. The algorithm packs the segments either to top or bottom depending on which side can result in a shorter wire length in detailed routing. For the wire segments preferred to be packed to the bottom, the packing algorithm iteratively finds a segment that can be assigned to the lowest location and assigns it. The lowest location is determined by the crosstalk-safe spacings to the segments already packed to the bottom. Packing segments to the top can be done similarly.

In DPPA, the minimum crosstalk-safe spacing between adjacent pseudopins from their crosstalk constraints is calculated in a similar way as in CPPA, except that we now have exact data on neighboring nets. It may not be the same value as previously estimated in CPPA. Therefore, it is possible that the spacing requirement in some interval exceeds the available space although it did not happen according to the previous estimations in CPPA. If this happens, we will adjust our spacing estimation, redo the coarse assignments for the affected nets with the new spacing estimations, and redo the ordering and spacing assignment on the affected strips.

2) Alignment Algorithm: For a strip s with height h, if we have enough spacing for all the pseudopins in each interval but the ordering and spacing algorithm generates assignment that require height larger than h, we would apply the alignment algorithm to introduce jogs and align as many pseudopins as possible.

For a strip s that contains several intervals, our alignment algorithm aligns pseudopins one interval at a time. It starts from the most crowded interval and works on the intervals toward its left and right.

For pseudopins $p_1, p_2, \ldots, p_n$ from bottom to top on an interval I_x on boundary B_i, we developed an *optimal dynamic programming* algorithm that assigns the pin locations for p_1, $p_2, \ldots, p_n$ and maximizes the number of alignments between the pseudopins on I_x to the locations (other pins of the same net) that they want to be aligned.

We use ms_i to denote the minimum separation distance between p_i and p_{i+1}. For a pseudopin assignment σ that assigns pseudopin p_i to location loc, we define $AL_i^\sigma(\mathrm{loc})$ as the number of alignments for pseudopins $p_1, p_2, \ldots, p_i$. We define $ML_i(\mathrm{loc})$ the maximum of $AL_i^\sigma(\mathrm{loc})$ among all feasible assignments and $MA_i(\mathrm{loc}) = \{\sigma \mid AL_i^\sigma(\mathrm{loc}) = ML_i(\mathrm{loc})\}$. Please note that the maximum of $ML_n(\mathrm{loc})$ for all feasible locations of pseudopin n is the maximum number of alignments.

It is obvious that $ML_i(\mathrm{loc})$ can also be recursively calculated by $ML_i(\mathrm{loc}) = a_i(\mathrm{loc}) + \max\{ML_{i-1}(b) \mid b + ms_{i-1} \leq \mathrm{loc}\}$, where $a_i(\mathrm{loc})$ is the number of alignments on p_i when it is assigned to location loc. Our alignment algorithm basically

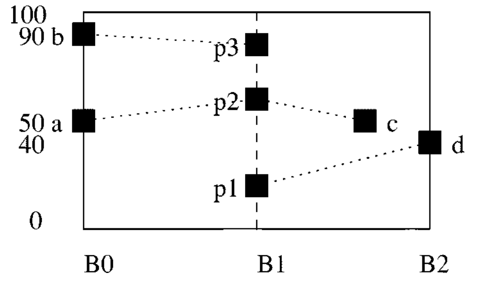


Fig. 7. Pseudopin alignments on a boundary.

uses this formula to calculate ML_i for $i = 1$ to n on all feasible locations.

We call a location loc_1 a redundant location for p_k if there exists a location loc_2 such that $\mathrm{loc}_1 > \mathrm{loc}_2$, and $ML_k(\mathrm{loc}_1) \leq ML_k(\mathrm{loc}_2)$. Because for any feasible assignment σ_1 that assigns p_k to loc_1, we can find an assignment σ such that it assigns p_k to loc_2 with greater or equal number of total alignments to σ_1. We can construct σ by combining σ_1 and $\sigma_2 \in MA_k(\mathrm{loc}_2)$ this way: $\sigma(i) = \sigma_2(i)$ for $i \leq k$ and $\sigma(i) = \sigma_1(i)$ for $i > k$. For any pseudopin p, we can restrict it to be assigned to nonredundant locations without losing the optimality. Please note if p_k has m nonredundant locations $\mathrm{loc}_{k,1} < \mathrm{loc}_{k,2} < \cdots < \mathrm{loc}_{k,m}$, we must have $ML_k(\mathrm{loc}_{k,1}) < ML_k(\mathrm{loc}_{k,2}) < \cdots < ML_k(\mathrm{loc}_{k,m})$. Therefore, to find the maximum ML_{i-1} below loc, we only need to find the maximum nonredundant location of p_{i-1} that is smaller than loc.

For pseudopin p_i, we can assume that the boundary B is partitioned to c_i intervals, $I_{i,1}, I_{i,2}, \ldots, I_{i,c_i}$ such that locations of the same interval have the same alignments and locations on adjacent intervals have different alignments. For example, in Fig. 7, we show pseudopin p_1 wants to be aligned with d at 40. The partition for p_1 is [0, 40), [40, 40], and (40, 100] with alignments zero, one, and zero, respectively.

We will show next if pseudopin p_{i-1} has a finite number m of nonredundant locations and p_i partitions the boundaries to t intervals, the number of nonredundant locations for pseudopin p_i is at most $m + t$. For an interval I, if there are x locations of the form $\mathrm{loc} + ms_{i-1}$ for some nonredundant location loc of p_{i-1}, the interval I, can be partitioned to $I_0, I_1, \ldots, I_x$ by these x points. Because the locations in each interval will have the same ML_i number, they are all redundant except for the lowest point on each interval. Therefore, interval I can contribute at most $x + 1$ nonredundant locations for p_i. Because there are at most m points that partition t intervals, the total number of nonredundant locations of p_i is at most $m + t$.

The above proof of finite nonredundant locations also gives us the procedure on how to compute them. We will use the example in Fig. 7 to demonstrate the calculation. We assume the minimum spacing is 20 for any pair of pseudopins. The nonredundant locations for pseudopin p_1 are 0 and 40 with $ML_1(0) = 0$ and $ML_1(40) = 1$. The pseudopin p_2 partitions the boundary to [0, 50), [50, 50], and (50, 100] with alignments zero, two, and zero, respectively. Therefore, we check on the locations $0 + 20$, 50, $40 + 20$ for nonredundant locations. The results are 20 and 50 with $ML_2(20) = 0$ and $ML_2(50) = 2$. Note that the location 60 is a redundant location of p_2 because $ML_2(60) = 1 < ML_2(50)$. The pseudopin p_3 partitions the boundary to [0, 90),

TABLE I
CROSSTALK NOISE ESTIMATION IN PPA (VIA COST)

	CPPA	noise	Noise distribution						avg.	run
		control	0.0-0.1	0.1-0.2	0.2-0.3	0.3-0.4	0.4-0.5	0.5-0.6	noise	time
mcc1	net	no	171	483	130	16	2	0	0.15	10.63
mcc1	id	no	351	433	18	0	0	0	0.11	26.09
mcc2	net	no	378	1422	2852	2001	459	6	0.26	80.73
mcc2	id	no	1224	2807	2611	466	10	0	0.19	315.06
s5378	net	no	1391	220	74	9	0	0	0.05	6.25
s5378	id	no	1405	234	51	4	0	0	0.04	12.27
s9234	net	no	1177	202	92	12	3	0	0.05	4.78
s9234	id	no	1259	187	38	2	0	0	0.04	8.83
s13207	net	no	3246	353	153	24	5	0	0.04	14.58
s13207	id	no	3367	286	113	13	2	0	0.03	34.46
s15850	net	no	3761	481	190	35	5	0	0.04	18.98
s15850	id	no	3928	401	123	15	5	0	0.03	53.05
s38417	net	no	10306	639	297	55	12	0	0.02	22.37
s38417	id	no	10911	354	41	3	0	0	0.01	40.36
s38584	net	no	13160	964	477	139	13	1	0.03	29.11
s38584	id	no	14147	516	82	9	0	0	0.01	64.85
mcc1	net	yes	315	465	22	0	0	0	0.12	12.82
mcc1	id	yes	400	390	12	0	0	0	0.10	36.78
mcc2	net	yes	1425	3848	1845	0	0	0	0.16	128.49
mcc2	id	yes	2236	3819	1063	0	0	0	0.14	784.78
s5378	net	yes	1476	209	9	0	0	0	0.03	16.96
s5378	id	yes	1518	170	6	0	0	0	0.03	20.58
s9234	net	yes	1322	156	8	0	0	0	0.03	9.58
s9234	id	yes	1364	118	4	0	0	0	0.03	15.02
s13207	net	yes	3475	273	33	0	0	0	0.02	24.31
s13207	id	yes	3581	186	14	0	0	0	0.02	67.15
s15850	net	yes	4051	382	39	0	0	0	0.03	35.96
s15850	id	yes	4237	225	10	0	0	0	0.02	93.15
s38417	net	yes	10617	612	80	0	0	0	0.02	25.97
s38417	id	yes	10803	466	40	0	0	0	0.01	53.99
s38584	net	yes	13761	893	100	0	0	0	0.02	34.61
s38584	id	yes	14074	624	56	0	0	0	0.01	95.95

$[90, 90]$, and $(90, 100]$ with alignments 0, 1, and 0, respectively. The nonredundant locations of p_3 are $20+20$, $50+20$, and 90 with $ML_3(40) = 0$, $ML_3(70) = 2$, and $ML_3(90) = 3$. If we assign p_3 to 90, $p2$ to 50 and p_1 to zero, we have an assignment with maximum alignments.

Our algorithm is further enhanced to allow weighted sums on the number of alignments such that we can have preferences to the alignments, which align pseudopins to real pins or pins on a previously processed boundary.

Note that the above alignment algorithm does not change the ordering of the pseudopins and the packing algorithm does not cause wire crossing on any segments. Therefore, misalignments of the pseudopins within a strip may be routed by just introducing jogs but not vias.

IV. EXPERIMENTAL RESULTS

We tested our algorithms on a 168-MHz SUN Ultra II. We use the NTRS'97 0.18-μm technology for the resistance and capacitance calculations.

We have two sets of test cases. The first set of test cases contains two test cases $mcc1$ and $mcc2$ of MCM designs. They were scaled down by a factor of 90.90 such that the original 75-μm pitch in MCM design is scaled to 1.5 times the minimum width (0.22 μm) plus the minimum spacing (0.33 μm) in this technology. The driver resistance in each net is set to 1800 Ω in this set of test cases.

The second set of test cases contains six standard-cell designs: s5378, s9234 s13207, s15850, s38417, and s38584. The placements of these test cases were placed by GORDIAN/DOMINO [13], [21].[8] The driver resistance in each net is set between 8550 and 1425 Ω according to the net length.

We use the global router MINOTAUR [9] to obtain the multilayer global routing solutions for all the test cases.

Each test case is run with different setups on the PPA algorithms, noise control, and cost functions. For the experiments with noise control, the noise constraint is set to 0.3 V_{DD} for each net.

For the test cases with combined length and via cost function, we use ten pitches as the length unit and the cost for h length units and v vias is $h + v$.

Tables I and II show the distribution of the estimated crosstalk noise on each test case with via cost function and combined cost function, respectively. The second column shows which CPPA algorithm is used (net for net-by-net and id for iterative deletion). The columns under "Noise distribution" give the number of nets falls in each range. For example, the column 0.2–0.3 shows the number of nets with crosstalk noise between 0.2–0.3 V_{DD}. The arithmetic average of the crosstalk noise of all nets are shown in the column "avg. noise." The last column shows the runtime measured in seconds.

If the crosstalk noise constraints are not considered, the average noise of $mcc1$ and $mcc2$ is 0.11–0.26 V_{DD} with up to 36% of nets that have noise larger than 0.3 V_{DD}. The average noise for those standard-cell test cases is much lower (0.01–0.05 V_{DD}).

[8]The test cases that we obtained only contain net lists without cell library information. The cell geometry and pin locations used in placement are generated by using a Mississippi State University 0.8-μm Standard Cell Library. We then shrunk them to approximate the cells in the 0.18-μm technology.

TABLE II
CROSSTALK NOISE ESTIMATION IN PPA (COMBINED COST)

	CPPA	noise control	Noise distribution 0.0-0.1	0.1-0.2	0.2-0.3	0.3-0.4	0.4-0.5	0.5-0.6	avg. noise	run time
mcc1	net	no	251	487	59	5	0	0	0.13	9.25
mcc1	id	no	308	461	33	0	0	0	0.12	20.57
mcc2	net	no	811	1442	2279	2033	553	0	0.25	69.68
mcc2	id	no	1154	2260	2218	1255	230	1	0.21	198.31
s5378	net	no	1363	225	94	11	1	0	0.05	6.26
s5378	id	no	1376	230	81	6	1	0	0.05	12.43
s9234	net	no	1164	222	87	10	3	0	0.05	4.76
s9234	id	no	1185	222	70	7	2	0	0.05	8.7
s13207	net	no	3247	372	131	27	4	0	0.04	15.2
s13207	id	no	3293	337	130	18	3	0	0.03	31.36
s15850	net	no	3758	488	187	36	3	0	0.04	19.18
s15850	id	no	3816	453	174	23	6	0	0.04	48.18
s38417	net	no	10344	622	283	51	9	0	0.02	21.93
s38417	id	no	10362	606	296	37	8	0	0.02	35.29
s38584	net	no	13131	950	505	147	20	1	0.03	29.22
s38584	id	no	13318	940	397	90	9	0	0.02	49.26
mcc1	net	yes	316	461	33	0	0	0	0.12	18.38
mcc1	id	yes	382	407	13	0	0	0	0.11	37.39
mcc2	net	yes	1493	3720	1905	0	0	0	0.16	181.23
mcc2	id	yes	2033	3869	1216	0	0	0	0.14	503.09
s5378	net	yes	1478	200	16	0	0	0	0.03	16.34
s5378	id	yes	1507	173	14	0	0	0	0.03	23.31
s9234	net	yes	1313	161	12	0	0	0	0.03	9
s9234	id	yes	1347	128	11	0	0	0	0.03	12.82
s13207	net	yes	3481	267	33	0	0	0	0.02	27.93
s13207	id	yes	3510	238	33	0	0	0	0.02	51.52
s15850	net	yes	4056	375	41	0	0	0	0.03	34.39
s15850	id	yes	4089	354	29	0	0	0	0.03	69.64
s38417	net	yes	10674	568	67	0	0	0	0.02	25.43
s38417	id	yes	10740	517	52	0	0	0	0.02	42.1
s38584	net	yes	13746	896	112	0	0	0	0.02	35.21
s38584	id	yes	13956	722	76	0	0	0	0.02	60.3

The noise distributions between MCM test cases and standard-cell test cases are quite different because the wire length distributions are quite different. For our MCM test cases, only 3 out of 802 nets in $mcc1$ and 81 out of 7118 nets in $mcc2$ are local nets that do not cross any tile boundary. On the other hand, most of the nets in these standard-cell designs are short local nets. 41% to 72% of nets that are local nets that do not cross any tile boundary. These short local nets usually have much less noise than global long nets. They are treated as no noise (0 V_{DD} when we calculate the noise distribution and noise average) in the PPA noise estimation. As a result,the estimated average noise of these standard-cell test cases are much lower than our MCM test cases.

If the crosstalk noise constraints are considered, the average noise in $mcc1$ and $mcc2$ is reduced to 0.10–0.16 V_{DD} and the average noise of those standard-cell test cases is reduced to 0.01–0.03 V_{DD}. Both of our CPPA algorithms successfully reduce all the noise values to values that are smaller than 0.3 V_{DD}. Overall, the noise control by iterative deletion algorithm is better, but the runtime is also longer. The difference in noise distributions using different cost functions in CPPA is not very significant.

The runtime of combined cost CPPA is comparable to the runtime of using via cost only because the selection of larger granularity when we calculate the wire length in CPPA [$\beta = 1$ in the $O(\beta dV)$ time algorithm]. In some test cases, the ones with combined cost are even faster than their via cost only counterparts.

We use an internally developed multilayer gridless detailed router based on the gridless routing engine as described in [5]–[7] to route the above examples. This detailed router is still under refinement; it can do a net-by-net routing, but can not do ripup and reroute at this point. The routing results are shown in Tables III and IV.

Please note that net counts in Tables III and IV count all the local nets and subnets within each tile. If a net spans n tiles, each of its n subnets is counted as one in the net counts.

The estimated number of vias in PPA are lower bound estimations,. The detailed router may use more vias if it makes more turns to route the nets. In PPA via estimation, we do not include the via estimation for local nets that could be routed within a single tile. However, the via counting in detail routing includes all the vias. This explains the big differences between the estimated and routed via counts.

The results of PPA are highly routable, even though no ripup and reroute is performed. The completion rates of different cases are 93%–99%. The average vias per net is only 0.7–1.4 in the MCM test cases and 1.0–1.7 in the standard-cell test cases.

The column "PPA wire length" reports the total Manhattan distance between pins after the DPPA. The column "DR wire length" reports the total wire length after the detailed routing. We can see the wire length in PPA is very close (within 4%) to the DR wire length. This shows that most of the detailed nets are routed without detours. We can see slight wire length reduction in the test case $mcc1$ if the combined cost is used. There is sizable wire length reduction (9%–10%) in $mcc2$ if the combined cost is used. The reason why we can get more wire length reduction in $mcc2$ than $mcc1$ is probably because we have smaller tiles in $mcc1$. Thus, less variations in wire lengths and less room for wire length reduction. Because most of the nets in our standard-cell test cases are local nets that do not have pseudopins

TABLE III
DETAILED ROUTING RESULTS (VIA COST)

	CPPA	noise control	Total nets	DR routed nets(%)	PPA est. vias	DR vias	PPA wire length(mm)	DR wire length(mm)
mcc1	net	no	12941	12876(99.50)	7414	9459	341.20	341.58
mcc1	id	no	12941	12918(99.81)	7378	9120	338.44	338.68
mcc2	net	no	55403	53737(97.03)	35542	47822	5423.37	5432.92
mcc2	id	no	55403	54776(98.87)	35430	44487	5394.07	5399.18
s5378	net	no	7000	6709(95.84)	4319	7009	81.52	84.08
s5378	id	no	7000	6767(96.67)	4269	7102	82.45	84.94
s9234	net	no	5823	5679(97.53)	3626	6086	58.33	60.35
s9234	id	no	5823	5734(98.47)	3623	5969	59.66	61.40
s13207	net	no	12602	11711(92.93)	8755	17011	180.82	185.84
s13207	id	no	12602	11788(93.54)	8504	17317	183.86	188.83
s15850	net	no	14644	13714(93.65)	10053	21364	224.54	232.00
s15850	id	no	14644	13789(94.16)	9596	21734	228.73	236.23
s38417	net	no	30010	29374(97.88)	14462	42151	481.12	495.52
s38417	id	no	30010	29399(97.96)	14433	41651	490.38	504.22
s38584	net	no	39060	38215(97.84)	17013	55215	661.86	684.03
s38584	id	no	39060	38371(98.24)	16951	54972	684.66	705.25
mcc1	net	yes	12941	12667(97.88)	7464	11403	341.40	341.93
mcc1	id	yes	12941	12713(98.24)	7426	11243	339.69	340.12
mcc2	net	yes	55403	52811(95.32)	40688	65044	5479.46	5495.27
mcc2	id	yes	55403	53198(96.02)	40144	64724	5419.49	5503.37
s5378	net	yes	7000	6488(92.69)	4510	7435	83.80	86.31
s5378	id	yes	7000	6530(93.29)	4476	7664	85.05	87.46
s9234	net	yes	5823	5619(96.50)	3717	6265	59.87	61.61
s9234	id	yes	5823	5642(96.89)	3653	6299	61.87	63.52
s13207	net	yes	12602	11746(93.21)	9070	18257	183.09	188.59
s13207	id	yes	12602	11731(93.09)	8727	18744	187.26	192.50
s15850	net	yes	14644	13532(92.41)	10556	22143	227.93	235.75
s15850	id	yes	14644	13503(92.21)	10069	22583	232.44	239.62
s38417	net	yes	30010	29257(97.49)	14286	42435	487.42	501.11
s38417	id	yes	30010	29312(97.67)	14214	42923	506.52	520.85
s38584	net	yes	39060	38102(97.55)	16673	55209	668.69	689.50
s38584	id	yes	39060	38183(97.75)	16609	56846	712.40	733.62

TABLE IV
DETAILED ROUTING RESULTS (COMBINED COST)

	CPPA	noise control	Total nets	DR routed nets(%)	PPA est. vias	DR vias	PPA wire length(mm)	DR wire length(mm)
mcc1	net	no	12941	12836(99.18)	7484	10859	326.88	327.36
mcc1	id	no	12941	12861(99.38)	7378	10696	327.33	327.66
mcc2	net	no	55403	54087(97.62)	36356	51501	4979.53	4988.44
mcc2	id	no	55403	54369(98.13)	35802	51920	4975.54	4982.63
s5378	net	no	7000	6715(95.93)	4727	6979	80.19	82.96
s5378	id	no	7000	6724(96.06)	4645	6975	80.20	82.75
s9234	net	no	5823	5697(97.84)	3803	6088	58.16	60.14
s9234	id	no	5823	5733(98.45)	3803	6080	58.33	60.18
s13207	net	no	12602	11717(92.98)	8908	16952	178.25	183.46
s13207	id	no	12602	11748(93.22)	8612	16935	178.55	183.64
s15850	net	no	14644	13645(93.18)	10040	20703	220.93	228.44
s15850	id	no	14644	13670(93.35)	9671	21038	220.88	228.78
s38417	net	no	30010	29316(97.69)	16311	41728	475.13	490.53
s38417	id	no	30010	29393(97.94)	15759	41919	474.50	489.10
s38584	net	no	39060	38149(97.67)	18175	55397	657.88	680.65
s38584	id	no	39060	38235(97.89)	17836	55179	657.38	679.34
mcc1	net	yes	12941	12647(97.73)	7604	13569	329.36	330.03
mcc1	id	yes	12941	12650(98.75)	7510	13880	329.66	330.22
mcc2	net	yes	55403	52029(93.91)	42968	74438	5031.27	5048.75
mcc2	id	yes	55403	52256(94.32)	42022	75094	5018.54	5031.85
s5378	net	yes	7000	6509(92.99)	4739	7235	82.16	84.44
s5378	id	yes	7000	6508(92.97)	4623	7319	82.18	84.16
s9234	net	yes	5823	5580(95.83)	3878	6330	59.73	61.62
s9234	id	yes	5823	5606(96.27)	3857	6272	59.81	61.49
s13207	net	yes	12602	11733(93.10)	9074	18133	179.75	185.32
s13207	id	yes	12602	11753(93.26)	8862	18309	179.89	185.42
s15850	net	yes	14644	13586(92.78)	10499	21921	222.93	230.58
s15850	id	yes	14644	13506(92.23)	10199	22152	223.12	230.72
s38417	net	yes	30010	29317(97.69)	15255	42121	479.73	493.50
s38417	id	yes	30010	29361(97.84)	15030	42484	479.70	493.05
s38584	net	yes	39060	38111(97.57)	17247	55214	663.73	684.33
s38584	id	yes	39060	38228(97.87)	17180	55424	662.82	682.72

and, therefore, not affected by PPA, we can only see small (up to 7%) wire length reductions in these standard-cell test cases.

Tables V and VI show the distribution of crosstalk noise *after* detailed routing. From the detailed routed results, we do a two-dimensional extraction to find out the line resistance, the line capacitance and coupling capacitance for all the nets (including local nets, therefore, the noise in local nets is calculated in this step). We then plug in these data to the same noise calculation

TABLE V
CALCULATED CROSSTALK NOISE AFTER DETAILED ROUTING (VIA COST)

	CPPA	noise	Noise distribution						avg.
		control	0.0-0.1	0.1-0.2	0.2-0.3	0.3-0.4	0.4-0.5	0.5-0.6	noise
mcc1	net	no	125	557	119	1	0	0	0.15
mcc1	id	no	234	552	16	0	0	0	0.13
mcc2	net	no	520	2349	3683	565	1	0	0.22
mcc2	id	no	912	3417	2681	108	0	0	0.18
s5378	net	no	1316	331	47	0	0	0	0.06
s5378	id	no	1305	345	44	0	0	0	0.07
s9234	net	no	1162	281	35	0	0	0	0.06
s9234	id	no	1189	265	24	0	0	0	0.06
s13207	net	no	3090	640	47	1	0	0	0.06
s13207	id	no	3102	636	40	0	0	0	0.06
s15850	net	no	3476	905	88	2	0	0	0.07
s15850	id	no	3514	907	50	0	0	0	0.07
s38417	net	no	9354	1822	132	1	0	0	0.06
s38417	id	no	9413	1803	92	1	0	0	0.06
s38584	net	no	11796	2712	242	4	0	0	0.06
s38584	id	no	11939	2635	178	2	0	0	0.06
mcc1	net	yes	274	515	13	0	0	0	0.12
mcc1	id	yes	326	468	7	0	0	0	0.11
mcc2	net	yes	1348	4427	1343	0	0	0	0.15
mcc2	id	yes	1752	4459	907	0	0	0	0.14
s5378	net	yes	1369	316	9	0	0	0	0.06
s5378	id	yes	1369	320	5	0	0	0	0.06
s9234	net	yes	1240	235	3	0	0	0	0.06
s9234	id	yes	1249	229	0	0	0	0	0.06
s13207	net	yes	3134	612	31	1	0	0	0.06
s13207	id	yes	3167	587	24	0	0	0	0.06
s15850	net	yes	3608	810	53	0	0	0	0.06
s15850	id	yes	3633	797	40	1	0	0	0.06
s38417	net	yes	9442	1784	82	1	0	0	0.06
s38417	id	yes	9428	1798	83	0	0	0	0.06
s38584	net	yes	11866	2707	178	3	0	0	0.06
s38584	id	yes	11868	2729	156	1	0	0	0.06

TABLE VI
CALCULATED CROSSTALK NOISE AFTER DETAILED ROUTING (COMBINED COST)

	CPPA	noise	Noise distribution				avg.
		control	0.0-0.1	0.1-0.2	0.2-0.3	0.3-0.4	noise
mcc1	net	no	182	565	55	0	0.14
mcc1	id	no	217	559	26	0	0.13
mcc2	net	no	843	2368	3110	797	0.21
mcc2	id	no	1091	3106	2511	410	0.19
s5378	net	no	1313	333	47	1	0.07
s5378	id	no	1310	335	49	0	0.07
s9234	net	no	1163	276	39	0	0.07
s9234	id	no	1158	295	25	0	0.07
s13207	net	no	3123	604	49	2	0.06
s13207	id	no	3093	646	37	2	0.06
s15850	net	no	3546	835	89	1	0.07
s15850	id	no	3511	883	75	2	0.07
s38417	net	no	9377	1784	147	1	0.06
s38417	id	no	9274	1880	154	1	0.06
s38584	net	no	11810	2694	246	4	0.06
s38584	id	no	11792	2711	248	3	0.06
mcc1	net	yes	266	515	21	0	0.12
mcc1	id	yes	326	462	14	0	0.11
mcc2	net	yes	1565	4428	1125	0	0.15
mcc2	id	yes	1939	4431	748	0	0.14
s5378	net	yes	1358	327	9	0	0.06
s5378	id	yes	1398	284	12	0	0.06
s9234	net	yes	1242	234	2	0	0.06
s9234	id	yes	1273	203	2	0	0.05
s13207	net	yes	3177	566	34	1	0.06
s13207	id	yes	3182	576	20	0	0.06
s15850	net	yes	3603	812	56	0	0.06
s15850	id	yes	3581	843	47	0	0.06
s38417	net	yes	9371	1841	97	0	0.06
s38417	id	yes	9356	1844	109	0	0.06
s38584	net	yes	11877	2701	175	1	0.06
s38584	id	yes	11860	2726	166	2	0.06

formula in (1) to find out the exact noise in the layout. Tables V and VI verified that the noise control in the PPA can be carried out by the detailed routing with high fidelity. The noise is not as bad as estimated in Tables I and II. This is because our noise estimation in PPA is somewhat conservative on the total capacitance, which results in higher estimated noise.

For the MCM test cases, if no noise control is done in PPA, the average noise for each test case ranges 0.13–0.22 V_{DD} with up to 11% of nets exceeding the 0.3 V_{DD} noise budget after detailed routing in $mcc2$. With 0.3 V_{DD} noise constraints for all nets, the average noise for each net reduces 15%–31% with no noise violations. The cases that use iterative deletion for CPPA still give the best noise distribution after detailed routing.

For the standard-cell test cases, we can see most of our test cases do not have serious noise problems (the average noise is only 0.06–0.07 V_{DD} without noise control) because the majority of the nets in these test cases are short local nets. The improvements on noise distributions are moderate because our algorithms only tries to meet the noise constraints and the majority of local nets are not affected by PPA. In some test cases, there are up to three nets that violate the noise constraints. This is because the noise estimation in PPA only considers the effects from global nets and the estimated routing in PPA may not completely match the routing results by the detailed router.

Tables VII and VIII show the crosstalk noise for the global nets (nets cross at least on tile boundary) in the standard-cell test cases. We can see our PPA algorithms generate better noise distributions for those global net and reduce the average noise of global nets from 0.08–0.09 V_{DD} to 0.07–0.08 V_{DD} in the test cases.

V. CONCLUSION

In this paper, we presented a new approach for PPA with crosstalk noise control in multilayer gridless general-area

TABLE VII
CALCULATED GLOBAL NET CROSSTALK NOISE AFTER DETAILED ROUTING (VIA COST)

	CPPA	noise control	Noise distribution 0.0-0.1	0.1-0.2	0.2-0.3	0.3-0.4	avg. noise
s5378	net	no	618	287	45	0	0.09
s5378	id	no	616	294	40	0	0.09
s9234	net	no	595	235	33	0	0.08
s9234	id	no	631	208	24	0	0.08
s13207	net	no	1207	420	41	1	0.08
s13207	id	no	1239	397	33	0	0.08
s15850	net	no	1265	595	78	2	0.09
s15850	id	no	1303	594	43	0	0.09
s38417	net	no	2435	745	92	0	0.08
s38417	id	no	2597	629	46	0	0.07
s38584	net	no	2574	1260	169	1	0.09
s38584	id	no	2818	1106	80	0	0.08
s5378	net	yes	678	264	8	0	0.08
s5378	id	yes	683	265	2	0	0.08
s9234	net	yes	682	179	2	0	0.07
s9234	id	yes	678	185	0	0	0.07
s13207	net	yes	1245	396	27	1	0.07
s13207	id	yes	1281	374	14	0	0.07
s15850	net	yes	1383	515	42	0	0.08
s15850	id	yes	1434	477	29	0	0.08
s38417	net	yes	2553	678	41	0	0.07
s38417	id	yes	2592	639	41	0	0.07
s38584	net	yes	2725	1189	89	1	0.08
s38584	id	yes	2775	1168	61	0	0.08

TABLE VIII
CALCULATED GLOBAL NET CROSSTALK NOISE AFTER DETAILED ROUTING (COMBINED COST)

	CPPA	noise control	Noise distribution 0.0-0.1	0.1-0.2	0.2-0.3	0.3-0.4	avg. noise
s5378	net	no	617	287	45	1	0.09
s5378	id	no	619	286	45	0	0.09
s9234	net	no	593	233	37	0	0.08
s9234	id	no	598	240	25	0	0.08
s13207	net	no	1233	395	39	2	0.08
s13207	id	no	1214	424	29	2	0.08
s15850	net	no	1325	540	74	1	0.08
s15850	id	no	1336	538	64	2	0.08
s38417	net	no	2454	720	97	1	0.07
s38417	id	no	2416	752	103	1	0.08
s38584	net	no	2579	1252	171	2	0.09
s38584	id	no	2615	1226	162	1	0.09
s5378	net	yes	662	282	6	0	0.08
s5378	id	yes	707	236	7	0	0.07
s9234	net	yes	682	180	1	0	0.07
s9234	id	yes	702	159	2	0	0.07
s13207	net	yes	1269	374	25	1	0.07
s13207	id	yes	1302	354	13	0	0.07
s15850	net	yes	1410	487	43	0	0.08
s15850	id	yes	1406	499	35	0	0.08
s38417	net	yes	2525	695	52	0	0.07
s38417	id	yes	2544	683	45	0	0.07
s38584	net	yes	2714	1203	87	0	0.08
s38584	id	yes	2778	1153	73	0	0.08

routing. Our approach includes two steps: CPPA and DPPA. This two-step approach absorbs the obstacle considerations and can efficiently assign pseudopins to minimize total wire length and the estimated number of vias and control crosstalk noise. Our experimental results show that our PPA algorithm can generate suitable PPA to satisfy the crosstalk noise constraints after detailed routing and achieve high completion rate in detail routing.

APPENDIX
NP-COMPLETENESS PROOFS OF THE PPA AND CPPA PROBLEMS

In this appendix, we shall show the simple proofs of the NP-completeness of both PPA and CPPA problems by the reductions from the Hamiltonian path problem and the set partition problem, respectively.

A. PPA Feasibility Problem is NP-Complete

It is obvious that the PPA feasibility problem is in NP because we can verify the feasibility of a PPA solution in polynomial time. It remains to show that the PPA feasibility problem is NP-hard.

The idea of the proof is to reduce the Hamiltonian path problem, which is NP-complete (see [16]), to a single boundary PPA feasibility problem, i.e., a PPA problem instance with only one boundary. The Hamiltonian path problem asks whether a graph $G = (V, E)$ has a simple path $(v_{p_1}, v_{p_2}, \ldots, v_{p_n})$ such that $n = |V|$, $v_{p_i} \neq v_{p_j}$ if $i \neq j$ and $(v_{p_i}, v_{p_{i+1}}) \in E$.

Given a graph $G = (V, E)$, we can construct a single boundary PPA problem in polynomial time such that each vertex v in V corresponds to the pseudopin p_v. If there is an edge between v and u in G, the pseudopin p_u and p_v are noise-free and can be separated by minimum spacing ms. If there is no edge between v and u, the noise consideration requires p_u and p_v be separated at least by ds, where $ds > ms$. We set the height of the row as $(n-1)ms$. It is easy to show that if the PPA problem is feasible, there is a way to place pseudopins such that distance between any adjacent pseudopins is ms. Thus, a corresponding Hamiltonian path can be found. Similarly, if we have a Hamiltonian path in G, we can convert it to a feasible solution in the constructed PPA problem. Therefore, the Hamiltonian path problem can be reduced to the PPA problem. Because the Hamiltonian path problem is NP-complete, the PPA problem is NP-hard. The PPA problem is NP-complete because it is both in NP and NP-hard.

B. CPPA Feasibility Problem is NP-Complete

It is obvious that the CPPA feasibility problem is in NP because we can verify the feasibility in polynomial time. We shall show there is a trivial reduction from the set partition problem, which is NP-complete (see [16]) to the CPPA problem to prove the CPPA feasibility is NP-hard.

The set partition problem asks whether there is a partition of a set S of integers to A and $S - A$ such that $\sum_{x \in A} x = \sum_{x \in S-A} x$. From a set partition problem that $\sum_{x \in S} x = T$, we can form a CPPA instance on a single boundary with two intervals of height $T/2$. Each element $x \in S$ corresponds a pseudopin with spacing requirement x. It is trivial to show these two problems are equivalent. Thus, CPPA problem is NP-hard and NP-complete.

ACKNOWLEDGMENT

The authors would like to thank J. Fang and K.-Y. Khoo for providing the gridless detailed router DUNE and P. Madden for providing MINOTAUR global router and some global routing solutions.

References

[1] C.-C. Chang and J. Cong, "pseudopin assignment with crosstalk noise control," in *Proc. 2000 Int. Symp. Physical Design*, Apr. 2000, pp. 41–47.

[2] K. Chaudhary, A. Onozawa, and E. S. Kuh, "A spacing algorithm for performance enhancement and cross-talk reduction," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, Nov. 1993, pp. 697–702.

[3] H. H. Chen and C. K. Wong, "Wiring and crosstalk avoidance in multichip module design," in *Proc. IEEE Custom Integrated Circuits Conf.*, May 1992, pp. 28.6.1–28.6.4.

[4] J. Cong, "Pin assignment with global routing for general cell designs," *IEEE Trans. Computer-Aided Design*, vol. 10, pp. 1401–1412, Nov. 1991.

[5] J. Cong, J. Fang, and K.-Y. Khoo, "An implicit connection graph maze routing algorithm for ECO routing," in *Proc. ACM/IEEE Int. Conf. Computer-Aided Design*, Nov. 1999, pp. 163–167.

[6] ——, "DUNE: A multilayer gridless routing system with wire planning," in *Proc. Int. Symp. Physical Design*, Apr. 2000, pp. 12–18.

[7] ——, "Via design rule consideration in multilayer maze routing algorithms," *IEEE Trans. Computer-Aided Design*, vol. 19, pp. 215–223, Feb. 2000.

[8] J. Cong, L. He, A. B. Kahng, D. Noice, N. Shirali, and S. H.-C. Yen, "Analysis and justification of a simple, practical 2 1/2-D capacitance extraction methodology," in *Proc. 34th ACM/IEEE Design Automation Conf.*, June 1997, pp. 627–632.

[9] J. Cong and P. H. Madden, "Performance driven multilayer general area routing for PCB/MCM designs," in *Proc. 35th Design Automation Conf.*, June 1998, pp. 356–361.

[10] J. Cong, D. Z. Pan, and P. V. Srinivas, "Improved crosstalk modeling for noise constrained interconnect optimization," in *Proc. Asia South Pacific Design Automation Conf. 2001*, Jan. 2001, pp. 373–378.

[11] J. Cong and B. Preas, "A new algorithm for standard cell global routing," in *IEEE Int. Conf. Computer-Aided Design*, Nov. 1988, pp. 80–83.

[12] A. Devgan, "Efficient coupled noise estimation for on-chip interconnects," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1997, pp. 147–153.

[13] K. Doll, F. Johannes, and G. Sigl, "DOMINO: Deterministic placement improvement with hill-climbing capabilities," *IFIP Trans. A (Comput. Sci. Technol.)*, vol. A-1, pp. 91–100, Aug. 1991.

[14] T. Gao and C. L. Liu, "Minimum crosstalk switchbox routing," *Integr. VLSI J.*, vol. 19, pp. 161–180, Nov. 1995.

[15] ——, "Minimum crosstalk channel routing," *IEEE Trans. Computer-Aided Design*, vol. 15, pp. 465–474, May 1996.

[16] M. R. Garey and D. S. Johnson, *Computers and Intractability. A Guild to the Theory of NP-Completeness*. San Francisco, CA: Freeman, 1979.

[17] T. Hameenanttila, J. D. Carothers, and D. Li, "Fast coupled noise estimation for crosstalk avoidance in the MCG multichip module autorouter," *IEEE Trans. VLSI Syst.*, vol. 4, pp. 356–368, Sept. 1996.

[18] W.-C. Kao and T.-M. Parng, "Cross point assignment with global rerouting for general-architecture designs," *IEEE Trans. Computer-Aided Design*, vol. 14, pp. 337–348, Mar. 1995.

[19] H. Kawaguchi and T. Sakurai, "Delay and noise formulas for capacitively coupled distributed RC lines," in *Proc. Asia South Pacific Design Automation Conf.*, Feb. 1998, pp. 35–43.

[20] D. A. Kirkpatrick and A. L. Sangiovanni-Vincentelli, "Techniques for crosstalk avoidance in the physical design of high-performance digital systems," in *Proc. IEEE Int. Conf. Computer-Aided Design*, Nov. 1994, pp. 616–619.

[21] J. Kleinhans, G. Sigl, F. Johannes, and K. Antreich, "GORDIAN: VLSI placement by quadratic programming and slicing optimization," *IEEE Trans. Computer-Aided Design*, vol. 10, Mar. 1991.

[22] D. P. Lapotin, "Early assessment of design, packaging and technology tradeoffs," *Int. J. High Speed Electron.*, vol. 2, no. 4, pp. 209–233, 1991.

[23] J. K. Ousterhout, "Corner stitching: A data-structuring technique for VLSI layout tools," *IEEE Trans. Computer-Aided Design*, vol. CAD-3, no. 1, pp. 87–99, Jan. 1984.

[24] T. Sakurai, "Closed-form expressions for interconnection delay, coupling, and crosstalk in VLSIs," *IEEE Trans. Electron Devices*, vol. 40, pp. 118–124, Jan. 1993.

[25] T. Stohr, M. Alt, A. Hetzel, and J. Koehl, "Analysis, reduction and avoidance of crosstalk on VLSI chips," in *1998 Int. Symp. Physical Design*, Apr. 1998, pp. 211–218.

[26] H.-P. Tseng, L. Scheffer, and C. Sechen, "Timing and crosstalk driven area routing," in *Proc. 35th ACM/IEEE Design Automation Conf.*, June 1998, pp. 378–381.

[27] A. Vittal and M. Marek-Sadowska, "Crosstalk reduction for VLSI," *IEEE Trans. Computer-Aided Design*, vol. 16, pp. 290–298, Mar. 1997.

[28] T. Xue, E. S. Kuh, and D. Wang, "Post global routing crosstalk risk estimation and reduction," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, Nov. 1996, pp. 302–309.

[29] H. Zhou and D. F. Wong, "Global routing with crosstalk constraints," in *Proc. 35th ACM/IEEE Design Automation Conf.*, June 1998, pp. 374–377.

Chin-Chih Chang received the B.S. degree in computer science from National Taiwan University, Taipei, Twiwan, R.O.C., in 1989 and the M.S. degree in computer science from the State University of New York, Stony Brook, in 1993. He is currently working toward the Ph.D. degree in computer science at the University of California, Los Angeles.

His current research interests include VLSI CAD algorithms on performance-driven layout synthesis.

Jason Cong (F'00) received the B.S. degree in computer science from Peking University, Beijing, China, in 1985 and the M.S. and Ph.D. degrees in computer science from the University of Illinois at Urbana–Champaign in 1987 and 1990, respectively.

He is currently a Professor and Co-Director of the VLSI CAD Laboratory in the Computer Science Department of University of California, Los Angeles. He has been appointed a Guest Professor of Peking University since 2000. He served as the General Chair of the 1993 ACM/SIGDA Physical Design Workshop, the Program Chair and General Chair of the 1997 and 1998 Internation Symposium on Field Programmable Gate Arrays, respectively, Program Co-Chair of the 1999 International Symposium on Low-Power Electronics and Designs, and on the program committees of many conferences. He has authored or coauthored over 140 research papers. His current research interests include layout synthesis and logic synthesis for high-performance low-power VLSI circuits, design and optimization of high-speed VLSI interconnects, FPGA synthesis, and configurable architectures.

Dr. Cong received the Best Graduate Award from Peking University in 1985, the Ross J. Martin Award for Excellence in Research from the University of Illinois at Urbana-Champaign in 1989, the National Science Foundation Young Investigator Award in 1993, the Northrop Outstanding Junior Faculty Research Award from the University of California at Los Angeles in 1993, the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS Best Paper Award in 1995, the ACM SIGDA Meritorious Service Award in 1998, and an Semiconductor Research Corporation Inventor Recognition Award in 2000. He is an Associate Editor of IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS and the *ACM Transactions on Design Automation of Electronic Systems*.