

Pseudo Pin Assignment with Crosstalk Noise Control*

Chin-Chih Chang
UCLA Computer Science Department
Los Angeles, CA 90095
cchang@cs.ucla.edu

Jason Cong
UCLA Computer Science Department
Los Angeles, CA 90095
cong@cs.ucla.edu

ABSTRACT

This paper presents a new pseudo pin assignment (PPA) algorithm with *crosstalk noise control in multi-layer gridless general-area routing*. We propose a two-step approach that considers obstacles and minimizes the estimated number of vias under crosstalk noise constraints. Without crosstalk noise control in PPA, the average noise after detailed routing of our test cases is 0.13-0.22 V_{DD} with up to 8% 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 reduces 15%-31% to 0.11-0.15 V_{DD} with no crosstalk noise violations. Even without rip-up and reroute, the detailed routing completion rate is 95%-99% and the ratio of vias to nets is only 0.7-1.2.

1. 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 the 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 *pseudo pin* in this paper. The problem of determining the pseudo pin locations is called the *pseudo pin assignment problem*. Because pseudo pin assignment 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 pseudo pin assignment with via minimization and crosstalk noise control in hierarchical multi-layer gridless general-area routing.

In [10], a two-layer grid-based pseudo pin assignment algorithm is discussed. Their heuristic algorithm optimizes the

alignment of pseudo pins but does not consider crosstalk. There are some studies on controlling crosstalk noise in detailed or channel routing (e.g., [1] [2] [7] [8] [9] [12] [16]), or in global routing (e.g., [19] [20]). 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 [17], crosstalk noise is considered in a pseudo pin¹ assignment step. Their algorithm inserts pseudo pins on each boundary one by one with a priority ordering and then performs a space relaxation algorithm to further separate pseudo pins. Their greedy algorithm may lack a global view to align the pseudo pins of the same nets.

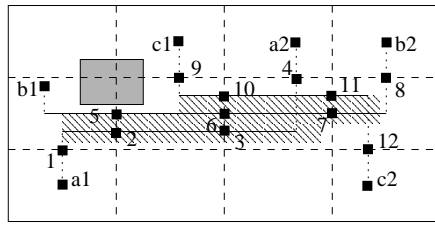
In this paper, we propose a new *pseudo pin assignment (PPA)* algorithm to control the crosstalk noise and minimize the number of vias in multi-layer gridless general area routing. Our algorithm absorbs the obstacle consideration by decomposing the tile boundaries into intervals and then solves the PPA problem in two steps: coarse pseudo pin assignment (CPPA) and detailed pseudo pin assignment (DPPA). In CPPA, each pseudo pin 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 vias and ensures that every interval has enough space for all the pseudo pins assigned to it. In DPPA, each pseudo pin is assigned to an exact location and crosstalk noise constraints must be satisfied. Our DPPA algorithm determines pseudo pin ordering and then aligns pseudo pins of the same net under crosstalk constraints.

2. PROBLEM FORMULATION

We are interested in the pseudo pin assignment (PPA) problem for a multi-layer gridless area routing system with obstacles. The inputs of the problem consist of a multi-layer 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

*This work is supported in part by DARPA/ETO under Contract DAAL01-96-K-3600 managed by the U.S. Army Research Laboratory, Semiconductor Research Corporation under Contract 98-DJ-605, and grants from Avant! Corporation and Fujitsu Laboratories at America under the California MICRO program.

¹In [17], pseudo pin is called crosspoint.

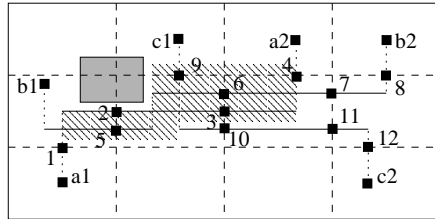


Vias = 6

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)

Figure 1: Impacts of pseudo pin assignment

can route each tile independently. Since these wire crossing locations act just like pins in detailed routing, we shall call them “*pseudo pins*” in this paper. On the other hand, we call the original pins “*real pins*” to distinguish them from pseudo pins. The problem of determining the locations of pseudo pins is called the *pseudo pin assignment problem*.

Assignments of pseudo pins may have significant impacts on the crosstalk noise and the number of vias in the final layout generated by the detailed router. For example, Figure 1 shows two pseudo pin assignments of the same global routing solution on 3×4 tiles. The tile boundaries are shown as dash lines. Pseudo pins are labeled 1-12; real pins are labeled $a1$, $a2$, $b1$, $b2$, $c1$, and $c2$; the grey areas are keep-out-regions. The possible detailed routings according to the pseudo pin assignment are also shown in the figure. The dotted lines represent wires on layer 1 and the 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 Figure 1(a), but decreases to 2 (tile widths) in Figure 1(b). The detour on net $b1 - b2$ is roughly 1 (tile height) in Figure 1(a) and 1.5 (tile height) in Figure 1(b). This example shows different PPA solutions can lead to considerably different via counts, wire lengths, and coupling between nets.

Our objectives of the pseudo pin assignment are to determine the locations of pseudo pins such that the estimated number of required vias is minimized and the crosstalk constraints are satisfied. Because crosstalk noise usually only needs to be controlled in a safe range and via minimization is usually desired, our objectives of PPA are to minimize the estimated number of vias under the crosstalk noise constraints. Note that minimizing the estimated number of vias

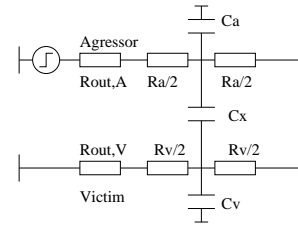


Figure 2: Crosstalk calculation

means more alignments on pseudo pins and less routing resources used by vias, thus generating more routable problem instances for detailed routing.

First, we explain how the crosstalk noise is estimated. The details of via estimation are explained in Section 3.1.

2.1 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 pseudo pins. If a pseudo pin is on the boundary between tiles T_1 and T_2 , the length of the wire segment 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 pseudo pins 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 [5] (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 [6] [11] [15] [16] [18].

In our implementation, we use a simple closed-form formula for two-terminal nets described in [16] to calculate the crosstalk noise on each wire segment.² According to [16], the peak crosstalk noise V_{noise} for the circuits in Figure 2 can be estimated by the following formula:

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

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

We assume the system clock is divided into n user-defined windows (time bucket) [13]. 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 user may specify whether there is noise concern between any pair of nets and

²If 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.

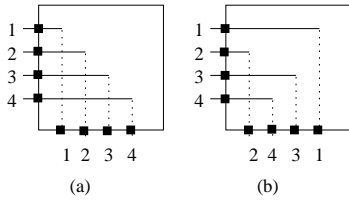


Figure 3: Net ordering on adjacent layers

in any window according to their logic switching behaviors. Our default assumption is that the noise between every net pairs should be considered in any window.

We use a simple assumption that 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.

2.2 Layer by Layer Approach

We observe that the assignment of pseudo pins in one layer has little affect on pseudo pin assignments on different layers. For example, Figure 3 shows that we can permute the pseudo pins on the vertical layer without changing the pseudo pin assignment 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 pseudo pins on the same layer.

Assigning pseudo pins one layer at a time can reduce the problem complexity and does not sacrifice too much solution quality. Furthermore, because each pseudo pin is confined on a single tile boundary, we do not need to work on the entire layer, assigning pseudo pins 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 pseudo pin assignment on a row of tiles in the later discussions.

3. PPA ALGORITHM

The crosstalk constrained pseudo pin assignment problem is an NP-hard problem even only considering a degenerated problem which determines if a feasible pseudo pin assignment exists on a single tile boundary. This can be proved by a simple reduction from the Hamiltonian path problem (similar to that in [8]).

Our pseudo pin assignment algorithm is a heuristic algorithm that consists of a tile boundary decomposition, a coarse pseudo pin assignment step, and a detailed pseudo pin assignment step. The algorithm flow is similar to the one in the pin assignment algorithm used in [3].

3.1 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 [14], are strips (rectangles) that form a partition on the empty space in a routing region such that no strips are horizontally adjacent to any other strips. In our algorithm, the rectangle objects are obstacles, real

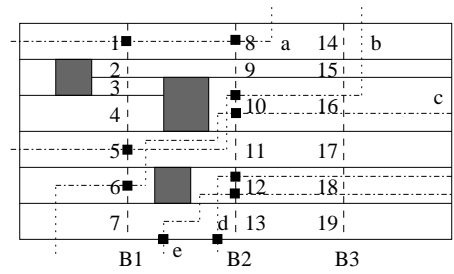


Figure 4: Tile boundary decomposition and via estimations

pins, or the projections of real pins from adjacent layers.³ Figure 4 shows an example of the maximum strips formed on a row of four tiles. The grey areas are rectangle objects, which are obstacles or real pins. 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 pseudo pin locations. We use the following approximation to simplify the estimation: we only consider obstacles (keep-out-regions) on the same layer and assume the space on the adjacent layers is always available for making connections.

If two pseudo pins 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 on 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 p_2 . Figure 4 shows several examples of the via estimation patterns on boundaries $B1$ and $B2$. The via estimations are 0, 2, 4, 1, and 3 for nets a , b , c , d , and e , respectively.

3.2 Coarse Pseudo Pin Assignment

Our CPPA algorithm first estimates a crosstalk-safe spacing for each pseudo pin and then assigns the pseudo pins to intervals with the objective of minimizing the estimated number of vias. A coarse pseudo pin assignment is feasible if all the intervals have enough space for the pseudo pins assigned to them. The CPPA problem is an NP-hard problem which can be proved by a simple reduction from the bin packing problem to a CPPA problem on a single boundary. The crosstalk-safe spacing for a pseudo pin is estimated by assuming that the pseudo pin is adjacent to a pair of pseudo pins which have the average capacitance, resistance, driver/receiver characteristics. We assume each pseudo pin has a noise budget that can be calculated from the noise constraints.⁴ If a pseudo pin has a noise budget B , with the estimated total capacitance, resistance, driver/receiver characteristics, we can calculate the maximum allowed coupling capacitance $C_x = (\frac{R_{out,A}}{R_{out,V} + R_v/2} C_a + C_v) B / V_{DD}$ on this pseudo pin from Equation 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 pseudo pin.

³In fact, each rectangle is expanded by half of the minimum spacing on that layer.

⁴In our implementation, we evenly distribute the noise constraint of each net to its pseudo pins according to their representing lengths. However, any clever algorithm which distributes noise budgets can be used.

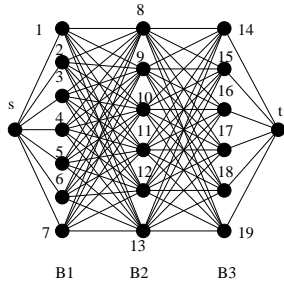


Figure 5: Coarse routing graph for CPPA

3.2.1 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 which are either real pins or pseudo pins. The edges in E connect vertex pairs which can reach each other without crossing a tile boundary.⁵

For example, Figure 5 shows the routing graph for a layout in Figure 4 with two connecting points s and t representing a net that needs to go through boundary $B1$ - $B3$. Each edge (v_1, v_2) in the routing graph is assigned a cost which is the estimated number of vias to connect pins on v_1 and v_2 .

It is easy to show that a coarse pseudo pin assignment for a net connecting from v to u corresponds to a path from v to u in the coarse routing graph, and the estimated number of vias of this net 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 via assignment.

3.2.2 Assignment of a Single Net

The CPPA for a single net can be solved by the shortest path algorithm for directed-acyclic-graph (DAG) in $O(V + E)$.⁶ Since the graph is very dense (i.e., $E = O(V^2)$), the complexity is $O(V^2)$. We have developed an algorithm to cut down the complexity of the shortest path algorithm to $O(dV + zV)$, where d is the maximum edge cost, and z is the maximum number of edges with zero-cost coming out from any vertex. Since d and z are small numbers ($d = 4$ and $z = 1$ in our formulation), we have a linear time algorithm in practice. The key idea of our algorithm is to avoid explicitly generating and visiting all the edges while maintaining the optimality of the shortest path algorithm.

We denote the distance (cost) between two vertices v_1, v_2 as $d(v_1, v_2)$. Given a vertex v , we denote the shortest distance from source s to v as $D(v)$. Remember 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 Figure 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

⁵In order to make uses of the intervals which are too short for assigning any pseudo pin, our routing graph has more vertices. For each interval I_v , we introduce two vertices v_{lo} and v_{hi} . If a pseudo pin 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 .

⁶For a set S , we will just use S for $|S|$ in the big-O notation when there is no confusion.

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 none-zero-cost edges will be pruned if the second condition is satisfied.

1. $D(w) + d < D(u)$.
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 . In the second case, any edge (u, x) which is not zero-cost is pruned by a similar reason. We say u is dominated by w and w dominates u if any of the above conditions is satisfied.

Our multi-stage shortest algorithm is basically a modification of Dijkstra's shortest path algorithm with pruning. We assume that the number of stages (the number of boundaries) is k , any edge cost c is an integer and $0 \leq c \leq d$, and the maximum number of edges with length zero coming out from any vertex is z . We use $D(v)$ to denote the shortest distance from source s to v . We use V^i to denote the vertex set in stage i (on Boundary i). We use Q_n^i to denote the vertex set in V^i whose shortest distance to s is n . Our algorithm starts from the source s and puts any reachable vertex v in stage i to Q_n^i if the current shortest distance from s to v is n . We go through the vertices of Q_n^i from smaller n to larger n . That is, as in Dijkstra's algorithm, we scan vertices with smaller shortest paths before vertices with larger shortest paths. Instead of visiting all the neighbors of any vertex v scanned and updating their shortest paths (done by procedure Relax), we only visit those neighbors connected by zero-cost edges and we only visit all the neighbors of v if v is not dominated by other vertices. If a vertex has a shortest path larger than any vertex by $d + 1$, none of its neighbors are visited because all the edges to them are pruned. The details of our multi-stage shortest path algorithm is shown in Figure 6.

The complexity of the algorithm is analyzed below. The initialization on Lines 1 to 9 takes $O(k^2d)$ time. The run time of Lines 10 to 19 is dominated by the loop in Lines 13-15 and the loop in Lines 16-18. Because there are at most zV zero cost edges and each one is visited at most once in Lines 13-15, the total time spends in Lines 13-15 is bounded by $O(zV)$. Because there are at most $d + 1$ vertices per stage (boundary) that are not dominated by other vertices, any vertex can appear at most $d + 1$ times in Line 17. Therefore, total time spends on Lines 16-18 is bounded by $O(dV)$. From the above discussions, the run time of the algorithm *Multi-Stage Shortest Path* is bounded by $O(dV + zV + k^2d)$. In fact, the $O(k^2d)$ time spends in line 1 to 5 can be further reduced to $O(kd)$ because we only need to keep $d + 1$ different cost groups in each stage. We can modify the *Relax* procedure to let it run in amortized constant time and use only $d + 1$ cost groups per stage. Therefore, the run time can be bounded by $O(dV + zV + kd)$. Since $k < V$, we have an $O(dV + zV)$ algorithm. We have the following theorem.

THEOREM 1. *Coarse pseudo pin assignment on a single net can be done in $O(dV + zV)$.*

ALGORITHM Multi-Stage Shortest Path

```

1. for  $i \leftarrow 1$  to  $k$ 
2.    $Q_\infty^i \leftarrow V^i$ 
3.    $MinCost^i \leftarrow \infty$ 
4.   for  $j \leftarrow 0$  to  $kd$ 
5.      $Q_j^i \leftarrow \emptyset$ 
6.    $Q_0^i \leftarrow \{s\}$ 
7.   foreach  $v \in V^1$ 
8.     Relax( $s, v, 0$ )
9.    $n \leftarrow 0$ 
10.  while  $D(t) > n$ 
11.    foreach  $i \in \{x \mid Q_n^x \neq \emptyset\}$  in increasing order
12.      if  $n \leq MinCost^{i+1} + d$ 
13.        foreach  $u \in Q_n^i$ 
14.          foreach  $v \in V^{i+1}$  and  $d(u, v) = 0$ 
15.            Relax( $u, v, i$ )
16.        foreach  $u \in Q_n^i$  not dominated by other vertices
17.          foreach  $v \in V^{i+1}$ 
18.            Relax( $u, v, i$ )
19.     $n \leftarrow n + 1$ 

```

```

Relax( $u, v, i$ )
1. if  $D(v) > D(u) + d(u, v)$ 
2.   remove  $v$  from  $Q_{D(v)}^{i+1}$ 
3.    $D(v) \leftarrow D(u) + d(u, v)$ 
4.   if  $D(v) < MinCost^{i+1}$ 
5.      $MinCost^{i+1} \leftarrow D(v)$ 
6.   insert  $v$  to  $Q_{D(v)}^{i+1}$ 

```

Figure 6: Multi-stage shortest path algorithm

3.2.3 Coarse Pseudo Pin Assignment Algorithms

Based on the efficient CPPA algorithm for a single net, we implemented two approaches to solve the coarse pseudo pin assignment problem. The first one is a *net-by-net* approach which 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 shortest-path algorithm in the previous section, we will not discuss its details. The second one is an *iterative deletion* approach (similar to the concept first used in global routing [3]), which works one boundary at a time and simultaneously assigns all the unassigned nets crossing the boundary. The algorithm iteratively picks the most crowded unassigned boundary to do the assignment. The details of the iterative deletion on a single boundary is described below.

1. For each pseudo pin of an unassigned net on the boundary, find all the vertices on the boundary that admit the shortest paths. Put an assignment of the pseudo pin to every vertex found.
2. Iteratively delete the assignments of pseudo pins until no interval is overflowed. Each deletion first finds the most crowded interval and then removes the assignment of the pseudo pin with the most number of undeleted alternative assignments.
3. For each pseudo pin p with an assignment not yet deleted, if the corresponding net can still be assigned with a shortest path found in Step 1, assign the net with the shortest path.
4. If all nets are assigned, stop. If the set of the unassigned nets is not changed since the last time it was checked in this step, stop. Otherwise, repeat Step 1.

If an unroutable net is detected, a rip-up and reroute algorithm on the coarse routing graph is applied to rip-up the

problematic boundary and re-route all the nets pass through that boundary. During the rip-up and reroute, we will make some cost adjustments and possibly some adjustments on crosstalk-safe spacing of some pseudo pins.

3.3 Detailed Pseudo Pin Assignment

The detailed pseudo pin assignment works consists of two steps: (i) determine the ordering and spacing of pseudo pins inside each strip, (ii) align the pseudo pins in each strip under crosstalk-safe spacing without changing ordering.

3.3.1 Ordering and Spacing Algorithm

Our ordering and spacing algorithm works on one strip at a time. It is a simple algorithm that assumes pseudo pins of one net are aligned as one single wire segment. 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 pseudo pins from their crosstalk constraints is calculated as done in coarse pseudo pin assignment, 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.

3.3.2 Alignment Algorithm

The alignment algorithm works one strips at a time. For each strip, we first check if the ordering and spacing algorithm has already generated a feasible pseudo pin assignment. If so, no further changes will be done. Otherwise, our alignment algorithm aligns pseudo pins one boundary at a time. It starts from the most crowded boundary and works on the boundaries toward its left and right.

For pseudo pins p_1, p_2, \dots, p_n from bottom to top on a boundary, we developed an *optimal dynamic programming* algorithm which finds the maximum number of alignments between the pseudo pins on the boundary and their neighboring pins of the same nets.

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

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

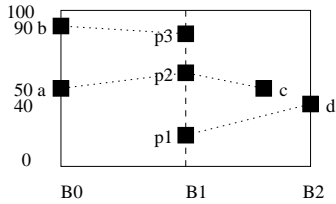


Figure 7: Pseudo pin alignments on a boundary

We call a location loc_1 a redundant location for p_k if there exists a location loc_2 such that $loc_1 > loc_2$, and $ML_k(loc_1) \leq ML_k(loc_2)$. 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 $\sigma_2 \in MA_{loc_2}$ this way: $\sigma(i) = \sigma_2(i)$ for $i \leq k$ and $\sigma(i) = \sigma_1(i)$ for $i > k$. For any pseudo pin p , we can restrict it to be assigned to non-redundant locations without losing the optimality. Please note if p_k has m non-redundant locations $loc_{k,1} < loc_{k,2} < \dots < loc_{k,m}$, we must have $ML_k(loc_{k,1}) < ML_k(loc_{k,2}) < \dots < ML_k(loc_{k,m})$. Therefore, to find the maximum ML_{i-1} below loc , we only need to find the maximum non-redundant location of p_{i-1} that is smaller than loc .

For pseudo pin p_i , we can assume that the boundary B is partitioned c_i intervals, $I_{i,1}, I_{i,2}, \dots, I_{i,c_i}$ such that locations of the same interval have the same alignments and locations on adjacent intervals have different alignments. For example, in Figure 7, we show pseudo pin 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 0, 1, and 0, respectively.

We will show next if pseudo pin p_{i-1} has a finite number m of non-redundant locations and p_i partitions the boundaries to t intervals, the number of non-redundant locations for pseudo pin p_i is at most $m+t$. For an interval I , if there are x locations of the form $loc + ms_{i-1}$ for some non-redundant location loc of p_{i-1} , the interval I , can be partitioned to I_0, I_1, \dots, 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 $x+1$ non-redundant locations for p_i . Because there are at most m points that partition t intervals, the total number of non-redundant locations of p_i is at most $m+t$.

The above proof of finite non-redundant locations also gives us the procedure on how to compute them. We will use the example in Figure 7 to demonstrate the calculation. We assume the minimum spacing is 20 for any pair of pseudo pins. The non-redundant locations for pseudo pin p_1 are 0 and 40 with $ML_1(0) = 0$ and $ML_1(40) = 1$. The pseudo pin p_2 partitions the boundary to $[0, 50]$, $[50, 50]$, and $(50, 100]$ with alignments 0, 2, and 0, respectively. Therefore, we check on the locations $0+20$, 50 , $40+20$ for non-redundant 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 pseudo pin p_3 partitions the boundary to $[0, 90]$, $[90, 90]$, and $(90, 100]$ with alignments 0, 1, and 0, respectively. The non-redundant 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, p_2 to 50 and p_1 to 0, we have an assignment

Table 1: Crosstalk noise estimation in PPA

		Noise distribution						avg. noise	run time
		0.1	0.2	0.3	0.4	0.5	0.6		
mcc1	nn	171	483	130	16	2	0	0.15	10.63
mcc1	in	351	433	18	0	0	0	0.11	26.09
mcc2	nn	378	1422	2852	2001	459	6	0.26	80.73
mcc2	in	1224	2807	2611	466	10	0	0.19	315.06
mcc1	ny	315	465	22	0	0	0	0.12	12.82
mcc1	iy	400	390	12	0	0	0	0.10	36.78
mcc2	ny	1425	3848	1845	0	0	0	0.16	128.49
mcc2	iy	2236	3819	1063	0	0	0	0.14	784.78

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 pseudo pins to real pins or pins on a previously processed boundary.

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

4. EXPERIMENTAL RESULTS

We tested our algorithms on a 168 MHz SUN Ultra II. We use the NTRS'97 0.18 μ m technology for the resistance and capacitance calculations. We have test cases *mcc1* and *mcc2* of MCM designs. They were scaled down by a factor of 90.90 such that the original 75 μ m pitch in MCM design is scaled to 1.5 times the minimum width (0.22 μ m) plus the minimum spacing (0.33 μ m) in this technology.

Each test case is run with different CPPA algorithms with and without noise control. For the experiments with noise control, the noise constraint is set to $0.3V_{DD}$ for each net. The driver resistance in each net is set to 1800 Ω . Table 1 shows the distribution of the estimated crosstalk noise on each test case. Each entry on the second column consists of two letters. The first letter is used to indicate which CPPA algorithm is used ('n' for net-by-net, 'i' for iterative-deletion). The second letter is used to indicate whether noise control is applied in PPA ('y' for yes, 'n' for no). For example, an entry "ny" in column two indicates the net-by-net CPPA algorithm is used and noise control is applied. A column of $0.d$ shows the number of nets with crosstalk noise between $0.(d-1)-0.d$ in each case. For example, the fifth column shows the number of nets with crosstalk noise between $0.2-0.3V_{DD}$. The arithmetic average of the crosstalk noise of all nets are shown in the ninth column. The last column shows the run time measured in seconds.

If the crosstalk noise constraints are not considered, the average noise is 0.11-0.26 V_{DD} with up to 34% of nets that have noise larger than $0.3V_{DD}$. If the crosstalk noise constraints are considered, the average noise is reduced to 0.10-0.16 V_{DD} and both algorithms successfully reduce all the noise to be smaller than $0.3V_{DD}$. Overall, the noise control by iterative deletion algorithm is better, but the run time is also longer. We use an internally developed multi-layer gridless detailed router based on the gridless routing engine presented in [4] to route the above examples. This detailed router is still under development, it can do a net-by-net routing, but can not do rip-up and reroute. The routing results are shown in Table 2. Note that the net counts in this table are the local nets within each tile. The estimated number of vias are lower bound estimations, and we do not include the via

Table 2: Detailed routing results

		Total nets	DR routed nets(%)	PPA est. vias	DR vias
mcc1	nn	12941	12876(99.50)	7414	9459
mcc1	in	12941	12918(99.81)	7378	9120
mcc2	nn	55403	53737(97.03)	35542	47822
mcc2	in	55403	54776(98.87)	35430	44487
mcc1	ny	12941	12667(97.88)	7464	11403
mcc1	iy	12941	12713(98.24)	7426	11243
mcc2	ny	55403	52811(95.32)	40688	65044
mcc2	iy	55403	53198(96.02)	40144	64724

Table 3: Crosstalk noise calculated after detailed routing

		Noise distribution						avg. noise
		0.1	0.2	0.3	0.4	0.5	0.6	
mcc1	nn	125	557	119	1	0	0	0.15
mcc1	in	234	552	16	0	0	0	0.13
mcc2	nn	520	2349	3683	565	1	0	0.22
mcc2	in	912	3417	2681	108	0	0	0.18
mcc1	ny	274	515	13	0	0	0	0.12
mcc1	iy	326	468	7	0	0	0	0.11
mcc2	ny	1348	4427	1343	0	0	0	0.15
mcc2	iy	1752	4459	907	0	0	0	0.14

estimation for 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 pseudo pin assignment are highly routable, even though no rip-up and reroute is performed. The completion rate is 95% – 99% and the ratio of the number of vias to routed nets is 0.7-1.2.

Table 3 shows the distribution of crosstalk noise *after* detailed routing. From the detailed routed results, we do a 2-D extraction to find out the line resistance, the line capacitance and coupling capacitance of the layout. We then plug in these data to the same noise calculation formula in Equation 1 to find out the exact noise in the layout. Table 3 verifies that the noise control in the pseudo pin assignment can be carried out by the detailed routing with high fidelity. The noise is not as bad as estimated in Table 1. This is because our noise estimation in PPA is somewhat conservative on the total capacitance, which results in higher estimated noise. If no noise control is done in PPA, the average noise is 0.13-0.22 V_{DD} with up to 8% of nets may exceed the 0.3 V_{DD} noise budget after detailed routing. With 0.3 V_{DD} noise constraints for all nets, the average noise reduces 15%-31% to 0.11-0.15 V_{DD} with no noise violations. The cases that use iterative deletion for CPPA still give the best noise distribution after detailed routing.

5. CONCLUSIONS

In this paper, we presented a new approach for pseudo pin assignment with crosstalk noise control in multi-layer grid-less general-area routing. We proposed a two-step approach to solve the problem. Our approach absorbs the obstacle considerations and can efficiently assign pseudo pins to minimize the estimated number of vias and control crosstalk noise. We developed an efficient algorithm for the assignment of a single net in the coarse pseudo pin assignment and an optimal dynamic programming algorithm for aligning pseudo pins on a boundary. Our experimental results show that our pseudo pin assignment algorithm can generate suitable pseudo pin assignment to satisfy the crosstalk noise constraints after detailed routing and achieve high completion rate in detail routing.

6. REFERENCES

- [1] K. Chaudhary, A. Onozawa, and E. S. Kuh. A spacing algorithm for performance enhancement and cross-talk reduction. In *1993 IEEE/ACM International Conference on Computer-Aided Design*, pages 697–702, 1993.
- [2] H. H. Chen and C. K. Wong. Wiring and crosstalk avoidance in multi-chip module design. In *Proceedings of the IEEE 1992 Custom Integrated Circuits Conference*, pages 28.6.1–28.6.4, 1992.
- [3] J. Cong. Pin assignment with global routing for general cell designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10(11):1401–1412, November 1991.
- [4] J. Cong, J. Fang, and K.-Y. Khoo. An implicit connection graph maze routing algorithm for ECO routing. In *Proceedings of ACM/IEEE International Conference on Computer Aided Design*, pages 163–167, 1999.
- [5] J. Cong, L. He, A. B. Kahng, D. Noye, N. Shirali, and S. H.-C. Yen. Analysis and justification of a simple, practical 2 1/2-D capacitance extraction methodology. In *Proceedings of 34th ACM/IEEE Design Automation Conference*, pages 627–632, 1997.
- [6] A. Devgan. Efficient coupled noise estimation for on-chip interconnects. In *Proc. Int. Conf. on Computer Aided Design*, pages 147–153, 1997.
- [7] T. Gao and C. L. Liu. Minimum crosstalk switchbox routing. *Integration, The VLSI Journal*, 19(3):161–180, November 1995.
- [8] T. Gao and C. L. Liu. Minimum crosstalk channel routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(5):465–474, May 1996.
- [9] T. Hameenanttila, J. D. Carothers, and D. Li. Fast coupled noise estimation for crosstalk avoidance in the MCG multichip module autorouter. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 4(3):356–368, September 1996.
- [10] W.-C. Kao and T.-M. Parng. Cross point assignment with global rerouting for general-architecture designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14(3):337–348, March 1995.
- [11] H. Kawaguchi and T. Sakurai. Delay and noise formulas for capacitively coupled distributed RC lines. In *Proc. Asia South Pacific Design Automation Conf.*, pages 35–43, 1998.
- [12] D. A. Kirkpatrick and A. L. Sangiovanni-Vincentelli. Techniques for crosstalk avoidance in the physical design of high-performance digital systems. In *Proceedings of 1994 IEEE International Conference on Computer Aided Design*, pages 616–619, 1994.
- [13] D. P. Lapotin. Early assessment of design, packaging and technology tradeoffs. *International Journal of High Speed Electronics*, 2(4):209–233, 1991.
- [14] J. K. Ousterhout. Corner stitching: a data-structuring technique for VLSI layout tools. *IEEE Transactions on Computer-Aided Design*, CAD-3(1):87–99, 1984.
- [15] T. Sakurai. Closed-form expressions for interconnection delay, coupling, and crosstalk in VLSIs. *IEEE Trans. on Electron Devices*, 40:118–124, 1993.
- [16] T. Stohr, M. Alt, A. Hetzel, and J. Koehl. Analysis, reduction and avoidance of crosstalk on VLSI chips. In *1998 International Symposium on Physical Design*, pages 211–218, 1998.
- [17] H.-P. Tseng, L. Scheffer, and C. Sechen. Timing and crosstalk driven area routing. In *Proceedings of 35th ACM/IEEE Design Automation Conference*, pages 378–381, 1998.
- [18] A. Vittal and M. Marek-Sadowska. Crosstalk reduction for VLSI. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 16(3):290–298, March 1997.
- [19] T. Xue, E. S. Kuh, and D. Wang. Post global routing crosstalk risk estimation and reduction. In *1996 IEEE/ACM International Conference on Computer-Aided Design*, pages 302–309, 1996.
- [20] H. Zhou and D. F. Wong. Global routing with crosstalk constraints. In *Proceedings of 35th ACM/IEEE Design Automation Conference*, pages 374–377, 1998.