# CROSSTALK NOISE CONTROL IN GRIDLESS GENERAL-AREA ROUTING*

*Chin-Chih Chang* *Jason Cong*
UCLA Computer Science Department, Los Angeles, California, U.S.A.

## ABSTRACT

This paper presents a new pseudo pin assignment algorithm for *crosstalk noise control* in *multi-layer gridless general-area routing*. It also tends to minimize the number of vias needed in detailed routing. Our approach consists of two steps: coarse pseudo pin assignment and detailed pseudo pin assignment. Coarse pseudo pin assignment uses the crosstalk noise constraints to estimate the spacing requirements for each pseudo pin and minimizes the number of vias. Detailed pseudo pin assignment determines the exact locations of pseudo pins with proper spacings to satisfy the crosstalk noise constraints.

## 1. INTRODUCTION

As VLSI technology advances, the minimum feature sizes of VLSI circuits are exponentially scaled down (from $250nm$ in 1997 to $70nm$ in 2009). On the other hand, the chip areas of VLSI systems are increased (from $300mm^2$ to $620mm^2$ respectively)[1]. These trends have the following impacts. First, devices are smaller and faster, but interconnects are more resistive with larger coupling capacitance, thus introducing relatively longer delays and more crosstalk noise in interconnects. Second, it enables much higher degree of on-chip integration, which leads to significant increase in the design complexity[2].

In order to cope the design complexity, reduce wire length, and increase chip density, hierarchical multi-layer general-area routing is used in both advanced IC, and MCM (Multichip Module) technologies. On the other hand, wire sizing and spacing were proposed to optimize interconnect performance and signal integrity, thus introducing gridless layout. The increased interconnect crosstalk noise has also become an important issue in designing low power and high performance VLSI circuits. Crosstalk noise can cause longer delays and even logical fault in circuits. It is important to control crosstalk noise in multi-layer gridless general-area routing.

There are some studies on controlling crosstalk noise in detailed routing or channel routing, e.g.,[3][4][5][6][7][8]. Although the crosstalk noise estimations during detailed routing can be accurate, the freedom to control crosstalk noise is restricted. In [9][10], crosstalk noise is estimated at the global routing level by considering track assignment during global routing. However, it is difficult to define "tracks" in gridless general-area routing. Furthermore, the present of obstacles (keep-out-regions) has made the track assignment problem a much harder problem. In [11], a grid-based MCM router with crosstalk avoidance is proposed. However, it is hard to pre-define "grid" in gridless routing. In [12], crosstalk noise is considered in a pseudo pin[1] assignment step. They solve the problem by greedily inserting pseudo pins on each boundary by a priority ordering and then performing a space relaxation algorithm to further separating pseudo pins. Their greedy algorithm may lack of a global view to align the pseudo pins.

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 consists of two steps: coarse pseudo pin assignment (CPPA) and detailed pseudo pin assignment (DPPA). It has the following advantages. (1) It can handle *gridless* layout with *obstacles*. (2) It can handle *crosstalk noise constraints*. (3) The number of required vias can be accurately estimated in CPPA. (4) The CPPA problem is a graph routing problem. Our two-step PPA algorithm changes a complex and difficult optimization problem to a well defined problem, thus making it possible to employee many existing good graph routing algorithms to solve the problem. (5) The DPPA problem is an obstacle free pseudo pin assignment problem, which makes it easier to be solved.

## 2. PROBLEM FORMULATION

We are interested in the pseudo pin assignment problem in a multi-layer gridless general-area routing. Pseudo pin assignment is an essential step to provide the necessary links between global routing and detailed routing in any hierarchical routing system. 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.

The inputs of the problem are a 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 rectangle tiles. For each net, the global routing solution determines which tiles and layers it should go through. The global router does not need to specify the exact net crossing locations on the tile boundary since it does not have much impacts on the global router's main objectives of selecting net topologies and reducing congestion. However, the detailed router needs these locations before it can route each individual tile. Since these net crossing

*This work is funded by DARPA/ETO under the contract DAAL01-96-K-3600 and a grant from Avant! Corporation under the California MICRO Program.

[1]In [12], pseudo pin is called crosspoint.

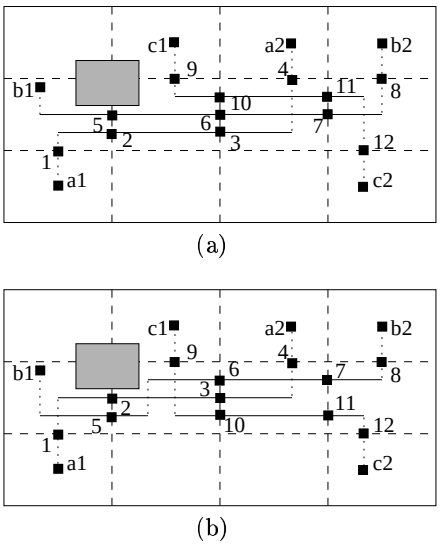


**Figure 1. Impacts of pseudo pin assignment**

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 $3 \times 4$ tiles. The tile boundaries are shown as dash lines. Pseudo pins are labeled 1-12; real pins are labeled $a1$-$c2$; the grey area is a keep-out-region. 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 readers can easily verify the difference of the number of vias, wire lengths, and coupling to adjacent 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. We choose these objectives because crosstalk noise is usually only needed to be controlled in a safe range and via minimization is usually desired. Furthermore, the number of required vias of a pseudo pin assignment is also a good indicator on the routability in detailed routing. If an assignment has fewer number of required vias, it means that more pseudo pins are aligned and less routing resources are required to make connections, which shall translate to higher routability in detailed routing.

The overall strategy to solve the pseudo pin assignment problem is to solve it one layer at a time to reduce the complexity. The assignment of pseudo pins on different layer has less impacts than the pseudo pins on the same layer. For example, Figure 2 shows that we can permute the pseudo pins on the vertical layer without changing the pseudo pins on the horizontal layer. We argue that assigning pseudo pins one layer at a time can reduce the problem complexity and does not sacrifice too much solution quality. Because each pseudo pin is confined on a single tile boundary, we do not need to work on the entire layer, assigning pseudo pins

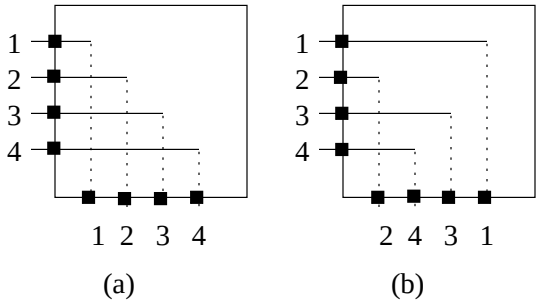


**Figure 2. Net ordering on adjacent layers**

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 in the later discussions.

The crosstalk constrained pseudo pin assignment problem is an NP-hard problem. In fact, it is NP-hard 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. Because of the NP-hardness of the problem, we only developed heuristic algorithm for solving the pseudo pin assignment problem.

## 3. CROSSTALK CALCULATION

This section discusses how we estimate the crosstalk noise and how to translate crosstalk noise constraints to spacing constraints. Our pseudo pin assignment algorithm can apply any crosstalk modeling to calculate the crosstalk noise, including those in [13][14][15][16][8], as long as the model only considers coupling from adjacent wires and allows us to calculate the crosstalk-safe spacing between wire segments. In our implementation, we use a close form formula to calculate the crosstalk noise. The formula is described in [8] which is an enhancement of a model described in [16].

We first discuss how the crosstalk noise is estimated. For each pseudo pin $p$, we use it to represent a wire segment in the net it associates with. If $p$ is on the boundary between tile $T_{p_1}$ and $T_{p_2}$, we use the the center-to-center distance between $T_{p_1}$ and $T_{p_2}$ to estimate the length of the wire segment. We denote the estimated length for the wire segment represented by pseudo pin $p$ as $l_p$. 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 and it needs to penalize coupling at the receiver more than coupling at the driver, we can use a weighted sum to calculate the crosstalk.

Given a pair of adjacent pseudo pins $p_v$ and pseudo pin $p_a$ separated with distance $d$, the coupling capacitance between the represented wire segments can be obtained by either an analytical formula or a table look up method. In our current implementation, we use a method as described in [17] to look up the capacitance.

According to [8], the peak crosstalk noise $V_{noise}$ for the circuits in Figure 3 can be estimated by the following formula.

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

The aggressor is driven by a step voltage source of $V_{DD}$ with intrinsic resistance of $R_{out,A}$; the victim is connected

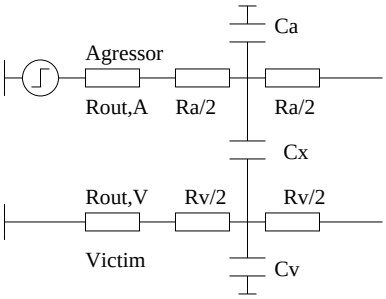


**Figure 3. Crosstalk calculation**

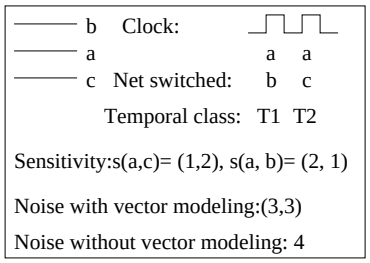


**Figure 4. Temporal separations of crosstalk noise**

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 $R_v$; the coupling capacitance is $C_x$.

We assume that $R_{out,A}$ and $R_{out,V}$ can be obtained from transistor characteristics. The line resistance $R$ of a line of width $w$ and length $l$ can be calculated by $R_{\square}l/w$, where $R_{\square}$ is the sheet resistance. The intrinsic capacitances for a line of width $w$ and length $l$ is estimated by $C^w l$, where $C^w$ is the unit length capacitance for a wire of width $w$. The coupling capacitance $C_x$ is obtained by the table look up as described above.

We now discuss how we calculate the minimum separation distances from crosstalk noise constraints. We assume that each net has a crosstalk noise constraint which is an upper bound on the interconnect coupling crosstalk noise this net can receive. For each net, the crosstalk noise constraint is distributed to each pseudo pin. In our implementation, the distribution is proportional to the representing length of each pseudo pin. For a pseudo pin $p_v$, if crosstalk noise constraint between $p_v$ and its adjacent pseudo pin $p_a$ is $V_{noise}$, we can calculate the maximum coupling capacitance allowed $C_x = \frac{V_{noise}}{V_{DD}}(\frac{R_{out,A}}{R_{out,V}+R_v/2}C_a + C_v)$ from Equation 1. After getting $C_x$, by a table look up (or a calculation from an analytical formula), we can find the minimum separation distance between $p_a$ and $p_v$ such that the coupling capacitance between them is less or equal to $C_x$. Similarly, we can find the minimum separation distance to satisfy the noise constraint of $p_a$. The minimum separation distance between $p_a$ and $p_v$ is the maximum of these two calculated distances and the minimum separation distance specified by design rules.

It is well known that the effective coupling capacitance $C_{eff}$ between two wires changes according to their circuit switching behaviors[5][7]. Assume the coupling capacitance between two wires is $C$, the effective coupling capacitance $C_{eff} = s_{ij}C$. In a simple model which only considers the switching directions, $s_{ij}$ is simply 0, 1, and 2 depends on the switching directions[5][7]. In models that consider input wave forms and switching directions, it may be as high as 5-6. We will use the effective coupling capacitance for the calculation of $C_x$ in the Equation 1.

We found that we can improve the usage of the crosstalk sensitivity modeling by using "temporal separations" in crosstalk noise. It is very common that certain logics are only triggered on certain clock edges. Therefore, it is possible to categorize the switching time of the circuits to $k$ temporal disjoint classes. We assume that the crosstalk noise in one temporal class do not propagate to other temporal disjoint classes. Under the above assumption, we can refine the crosstalk sensitivity as a k-tuple vector to represent sensitivities in the corresponding temporal classes. Using the temporal separation information can avoid over estimating crosstalk noise, thus allowing more aggressive designs.

A simple example is given in Figure 4 to demonstrate why using vectorized sensitivity may help. It shows three wires $a$, $b$, and $c$. Wire $a$ is adjacent to $b$ and $c$. Assume the circuits only switches on the first and second rising edges of the system clock, the sensitivity is defined as a two-tuple. Assume that the sensitivities between $a$ and $b$ is $(2, 1)$, and the sensitivity between $a$ and $c$ is $(1, 2)$; the crosstalk noise budget for wire $a$ is 3.5; the coupling capacitance from wires $b$ and $c$ to wire $a$ are both 1. Under the assumption that one unit of coupling capacitance results one unit of crosstalk noise, we can claculate the crosstalk noise on wire $a$. If temporal separation is considered, the crosstalk noise is $(3, 3)$ which is under the budget. However, if the temporal separation is not considered, the crosstalk is calculated as $2+2 = 4$ because the maximum sensitivity of $a$ to its adjacent wires are both 2. In this situation, the estimated crosstalk noise is over the budget, thus requiring further separation of wires which is not really necessary.

The vectorized sensitivity modeling is an enhancement on the crosstalk sensitivity modeling. It is always possible to have only one temporal class, thus reducing this formulation to the conservative traditional sensitivity modeling.

## 4. ALGORITHMS

Our pseudo pin assignment algorithm consists three major parts. First, tile boundaries are decomposed into intervals by maximum horizontal strips. Second, a coarse pseudo pin assignment algorithm is performed to assign pseudo pins to the intervals. Finally, a detailed pseudo pin assignment is applied to assign the exact pseudo pin locations. Such an approach of boundary decomposition follows by coarse assignment and detailed assignment is similar to the algorithm flow for the pin assignment algorithm used in [18].

### 4.1. Tile boundary decomposition

Given a row of tiles on a horizontal routing layer, we first decompose it by maximum horizontal strips. The *maximum horizontal strips* were first defined in [19], which means no strip has other strip immediately to its right or left.

Given a set of rectangle objects in a rectangle region, the maximum horizontal strips are formed by the boundaries of the region, the boundaries of the objects, and a set of horizontal shooting lines which shoot from both the top and bottom edges of the objects and stop when they hit object boundaries. In our algorithm, the rectangle objects are obstacles or real pins after expanded with half of the minimum separation distance specified by design rules.

A tile boundary is decomposed to a set of intervals by maximum horizontal strips. Each interval is an intersection

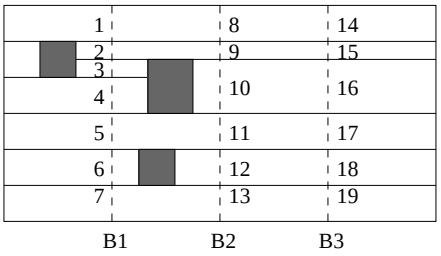


**Figure 5. Tile boundary decomposition**

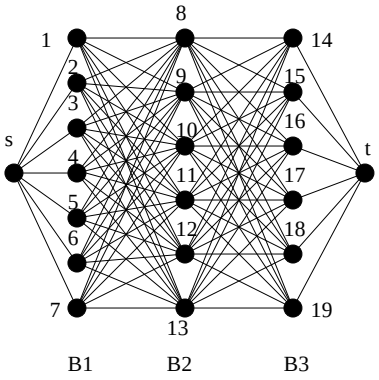


**Figure 6. Routing graph for CPPA**

of a maximum horizontal strip and a tile boundary. Figure 5 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. In the figure, tile boundaries $B1$-$B3$ are decomposed to intervals 1-19. The above tile boundary decomposition allows us to estimate the minimum number of required vias with great accuracy by just knowing which interval each pseudo pin is assigned to. Given a pair of pseudo pins assigned to two intervals, the estimation on the minimum number of vias required to connect these two pins is based on the geometry relation of these two intervals and the reserved layer routing model. The estimations can be classified by several patterns, which is discussed in [20].

### 4.2. Coarse pseudo pin assignment

Our coarse pseudo pin assignment (CPPA) algorithm assigns pseudo pin to the intervals. The objective is to minimize the estimated number of vias without overflowing any interval. An interval is overflowed if its height is less than the total estimated widths of the pseudo pins which are assigned to it. The estimated width of a pseudo pin is the sum of its wire widths and an estimated crosstalk-safe spacing which is calculated by its crosstalk constraint. A coarse pseudo pin assignment is feasible if all pseudo pins are assigned and no intervals are overflowed.

The coarse pseudo pin assignment can be solved as a graph routing problem. Given a tile boundary decomposition on a horizontal layer of a row of tiles, we can generate the routing graph $G = (V, E)$ where the vertex set $V$ consists of the intervals and the connection points which are either real pins, or pseudo pins which are on the other layers. The edge set $E$ consists of edges which connects any two vertices which can reach each other without crossing a tile boundary. For example, Figure 6 shows the routing

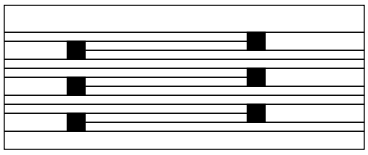

**Figure 7. Short-height strips**

graph for a layout in Figure 5 with two connecting points $s$ and $t$ representing a net needs to go through boundary $B1$-$B3$.

Each edge $(v_1, v_2)$ in the routing graph is assigned with a cost which is the estimated number of vias of connecting pins on $v_1$ and $v_2$. Given a coarse pseudo pin assignment for a net, the estimated number of vias is the sum of the edge costs of the path connecting the corresponding vertices of the assignments. Therefore, finding a feasible coarse pseudo pin assignment with minimum number of vias is equivalent to finding a set of routing paths with minimum total edge costs without overflowing the vertices. Since this problem is NP-hard, we solve it by heuristic algorithms.

It is possible to have some maximum horizontal strips with very small heights. Figure 7 shows an example of misaligned real pins and very short-height maximum horizontal strips. The short maximum horizontal strips may generate a lots of intervals which no pseudo pin can be assigned to. We address this problem by introducing a new type of vertices in the routing graph. For each interval vertex $v$, we will generate two extra vertices $v_{lo}$ and $v_{hi}$. Assigning a pseudo pin to $v_{lo}$ or $v_{hi}$ means the pseudo pin is aligned to the bottom or the top of $v$ respectively.

We have implemented two approaches to solve the coarse pseudo pin assignment problem. The first one is a *net-by-net* approach. Each net is assigned by a shortest path algorithm which finds the minimum cost assignment for pseudo pins. The second one is an *iterative deletion* approach, which simultaneous assigns all the unassigned nets in a boundary to avoid making premature assignments on some nets. The algorithm picks the most crowded unassigned boundary to do the assignment.

The details of the iterative deletion approach is described below.

1. For each unassigned net $n$ on the boundary $B$, run the shortest path algorithm to find the lowest cost assignments for net $n$. Find all the assignments on the boundary $B$ which can be part of the shortest paths which assign net $n$ with the lowest cost.
2. We iteratively delete an assignment found in step 1 until no intervals are overflowed. The selection on the assignment to be deleted is based on two criteria. First, it must come from an interval which is the most crowded. Second, among the assignments selected in the first step, the corresponding net of the selected assignment has the most number of alternative assignments.
3. For each unassigned net $n$ which has an assignment not yet deleted by iterative deletion, check if the net can be assigned by one of the remaining assignments with the same minimum cost as found in step 1. If this can be done, assign the net with the shortest path and mark net $n$ as assigned.
4. If all nets are assigned, stop. If the set of the unassigned nets does not change since last time it was checked in this step, stop. Otherwise, repeat step 1.

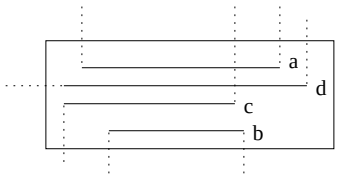


**Figure 8. Classification of segments in a strip**

It is possible that some nets can not find assignments by the above algorithms. If an unroutable net is detected, a rip-up and reroute algorithm should be applied to find the assignments.

### 4.3. Detailed pseudo pin assignment

The detailed pseudo pin assignment is done by assigning one maximum horizontal strip at a time. The algorithm which assigns the pseudo pins in one strip is highlighted below.

1. A simple packing algorithm is applied to determine the pseudo pin ordering by assuming all the pseudo pins of the same nets are aligned.
2. If the ordering determined in Step 1 can not be realized without breaking alignments of pseudo pins to satisfy crosstalk-safe spacing constraints, a refinement algorithm which re-assigns pseudo pins without changing the pseudo pin ordering on tile boundaries is applied. For each boundary, it maximizes the pseudo pin alignments on the boundary under the crosstalk-safe spacing constraints.
3. If the algorithm in Step 2 fails because the spacing constraints can not be satisfied, a rip-up and reroute algorithm is applied to rip up violating nets and reassign them.
4. When all the spacing violations are resolved, a space relaxation algorithm is applied. It calculates the remaining space in the strip which is not used for satisfying the spacing constraints. The remaining space is evenly distributed to increase the spacing between pseudo pins.

Since the rip-up and reroute algorithm is quite standard and the relaxation is very simple, we will not discuss the details of these algorithms. We will only briefly discuss the packing algorithm and the refinement algorithm. The packing algorithm is quite simple. For each net $n$ goes through the strip, assuming that the pseudo pins of net $n$ can be aligned on one line, we can use a wire segment to represent the pseudo pins.

The algorithm packs the segments either to top or bottom depending on which side can result a shorter wire length in detailed routing. For example, Figure 8 shows four segments in a strip. In this strip, segment $a$ will be preferred to be packed to the top, and segment $b$ to the bottom. For segments segments $c$ and $d$, they can be packed to either top or bottom.

For the wire segments preferred 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.

When the algorithm packs the wire segments, it assumes the floor and ceiling are separated far enough such the stack of wire segments on the bottom does not collide with the stack of wire segments on the top. The pseudo pin ordering on each boundary is determined according this assumption. If the height of the strip is tall enough to realize the above assumption, all the pseudo pins of the same nets assigned to the strip are aligned. However, it is possible that these two stacks of wire segments may collide. Therefore, we need the refinement algorithm to assign the pseudo pins on each boundary.

For each boundary, the refinement algorithm does not change the pseudo pin ordering on the boundary and is an *optimal dynamic programming* algorithm which finds the maximum number of alignments between the pseudo pins on the boundary and their adjacent pins of the same nets. It considers the pseudo pins on the boundary one by one in increasing order. For each pseudo pin, it constructs a table, where each table entry consists a possible total number of alignments below the pseudo pin (including itself) and the lowest possible location it can be assigned such that the corresponding number of alignments can be realized. For the pseudo pin at the bottom, the table can be easily constructed by checking the locations which can make itself aligned with pseudo pins of the same net. For any pseudo pin $p_t$ with minimum separation distance $d$ to the pseudo pin $p_b$ immediately below it, if the table of $p_b$ is known, the table for $p_t$ can be constructed by checking the locations which can make itself aligned, and the locations which are above the locations in the table of $p_b$ by distance $d$. Once the table for the pseudo pin at the top is known, we can find the locations for each pseudo pin to get maximum alignments. The refinement algorithm is enhanced by allowing weighted sums on the number of alignments to allow preferences on alignments to real pins and pins on a refined boundary.

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

## 5. EXPERIMENTAL RESULTS

We implemented our algorithm[2] and tested it on the $0.18um$ technology based on NTRS94. Because we have no access to a real design in this technology, we have scaled down a MCM test case *mcc*1 to do our experiments. The scaling factor is 90.90, which translate the original $75um$ pitch in MCM design to 1.5 times the minimum width ($0.22um$) plus the minimum spacing ($0.33um$) in this technology. The circuit size after the scaling is $482.35um \times 418.6um$. The global routing solution for this test case is generated by a tile-based global router, which divides the routing area to $16 \times 16$ tiles. The routing is done in four routing layers.

Each of our algorithms is run on the same example with crosstalk noise constraints and without crosstalk noise constraints. For the experiments without crosstalk noise constraints, we simply set the noise tolerance bound to an extremely high value such that the required minimum separation distances between pseudo pins are just the minimum spacing specified by the design rules. For the experiments with crosstalk noise constraints, the noise constraint is set

[2]Our implementation of our algorithm is a simplified version of the descriptions in this paper. We have not yet implemented the rip-up and reroute in the coarse pseudo pin assignment.

| noise range | w/o constraints | | with constraints | |
|---|---|---|---|---|
| | NbN | ID | NbN | ID |
| $(0, 0.05]$ | 8 | 30 | 29 | 42 |
| $(0.05, 0.1]$ | 38 | 79 | 139 | 192 |
| $(0.1, 0.15]$ | 122 | 195 | 351 | 368 |
| $(0.15, 0.2]$ | 192 | 282 | 255 | 187 |
| $(0.2, 0.25]$ | 225 | 171 | 28 | 13 |
| $(0.25, 0.3]$ | 121 | 37 | 0 | 0 |
| $(0.3, 0.35]$ | 57 | 7 | 0 | 0 |
| $(0.35, 0.4]$ | 27 | 1 | 0 | 0 |
| $(0.4, 0.45]$ | 8 | 0 | 0 | 0 |
| $(0.55, 0.5]$ | 3 | 0 | 0 | 0 |
| $(0.5, 0.55]$ | 1 | 0 | 0 | 0 |
| run time(s) | 169 | 254 | 261 | 502 |

**Table 1. Crosstalk in pseudo pin assignments**

| | w/o constraints | | with constraints | |
|---|---|---|---|---|
| | NbN | ID | NbN | ID |
| est. vias | 7394 | 7344 | 8216 | 8215 |
| routed vias | 11676 | 11452 | 14006 | 14113 |
| routed nets | 12779 | 12896 | 12907 | 12908 |
| completion | 0.987 | 0.997 | 0.997 | 0.997 |
| wire length(mm) | 337.88 | 337.87 | 345.76 | 344.48 |

**Table 2. Detailed routed results**

to be $0.25V_{DD}$ for each net. We do not have the sensitivity data, therefore, we set all the crosstalk sensitivity as 1. The driver resistance in each net is set as $1800\Omega$.

Table 1 shows the distribution of the crosstalk noise after each the pseudo pin assignment is applied. The run time of each algorithm is also shown at the bottom. The net-by-net algorithm is denoted as "NbN" in the tables, while the iterative deletion approach is denoted as "ID" in the table. Each row shows the number of nets which are in the noise range $(x, y]$, which means the noise is larger than $xV_{DD}$, but smaller or equal to $yV_{DD}$. We can see our algorithm can bring the crosstalk noise to the range within $0.25V_{DD}$ in this test case.

Table 2 shows the estimated lower bound of the number of vias and the routing results generated by a detailed router. The net count in this table is not the count of global net as in Table 1. It count the number of nets seen by the detailed router, i.e., if a global net goes through $k$ tiles, each subnet on each tile is counted as one net. The total number of detailed routing nets is 12941. The detailed router is a gridless multi-layer area router, which is still under development at UCLA. It has not yet included a rip-up and reroute module. The completion rate in detailed routing is good, but the number of vias is high. It is partly because the global routing solution we got does not penalize the usage of vias. The total number of layer changing by the global router is 6284, which means that it requires at least 6284 vias to finish the routing.

## 6. CONCLUSION

In this paper, we presented a new approach for crosstalk noise control in multi-layer gridless general-area routing. We proposed two steps approach to solve the problem. Our experimental results show that our pseudo pin assignment algorithm can generate suitable pseudo pin assignment to satisfy the crosstalk noise constraints and achieve high completion rate in detail routing.

## REFERENCES


[1] *The National Technology Roadmap for Semiconductors 1997 Edition.*

[2] J. Cong. Challenges and opportunities for design innovations in nanometer technologies. In *SRC Working Paper, http://www.src.org/prg_mgmt/frontier.dgw*, Dec. 1997.

[3] 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.

[4] Kamal Chaudhary, Akira Onozawa, and Ernest 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.

[5] Desmond A. Kirkpatrick and Alberto 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.

[6] Tong Gao and C.L. Liu. Minimum crosstalk switchbox routing. *Integration, The VLSI Journal*, 19(3):161–180, November 1995.

[7] Tong 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.

[8] 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.

[9] 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.

[10] Hai Zhou and D.F. Wong. Global routing with crosstalk constraints. In *35th Design Automation Conference*, pages 374–377, 1998.

[11] T. Hameenanttila, J.D. Carothers, and Donghui 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.

[12] Hsiao-Ping Tseng, Louis Scheffer, and Carl Sechen. Timing and crosstalk driven area routing. In *35th Design Automation Conference*, pages 378–381, 1998.

[13] A. Devgan. Efficient coupled noise estimation for on-chip interconnects. In *Proc. Int. Conf. on Computer Aided Design*, pages 147–153, 1997.

[14] T. Sakurai. Closed-form expressions for interconnection delay, coupling, and crosstalk in VLSIs. *IEEE Trans. on Electron Devices*, 40:118–124, 1993.

[15] 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.

[16] 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.

[17] Jason Cong, Lei He, Andrew B. Kahng, David Noice, Nagesh Shirali, and Steve H.-C. Yen. Analysis and justification of a simple, practical 2 1/2-d capacitance extraction methodology. In *Proceedings 1997. Design Automation Conference, 34th DAC*, pages 627–632, 1997.

[18] 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.

[19] 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.

[20] Chin-Chih Chang and Jason Cong. Crosstalk noises control in gridless general-area routing. Technical Report CSD-TR-980039, UCLA Computer Science Department, 1998.