Crosstalk Minimization using Wire Perturbations*

Prashant Saxena
Strategic CAD Labs
Intel Corporation
Hillsboro, OR 97124
psaxena@ichips.intel.com

C. L. Liu
Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan, R.O.C.
liucl@nthu.edu.tw

Abstract

We study the variation of the crosstalk in a net and its neighbors when one of its trunks is perturbed, showing that the trunk's perturbation range can be efficiently divided into subintervals having monotonic or unimodal crosstalk variation. We can therefore determine the optimum trunk location without solving any non-linear equations. Using this, we construct and experimentally verify an algorithm to minimize the peak net crosstalk in a gridless channel.

1 Introduction

With device geometry scaling, the crosstalk due to capacitive coupling between adjacent nets is becoming an increasingly major concern in high speed designs ([1]). Crosstalk minimization can be attempted either while routing the nets or as a postprocessing phase. Much of the previous work on this problem has taken the former approach. However, since the crosstalk in a wire depends on the relative positions of its neighbors (which may not yet have been put in place at the time the wire is being routed), every crosstalk-aware router is forced to use some rough estimate for the expected crosstalk. In contrast, although a postprocessing algorithm has less flexibility in moving the wires around, it can use accurate crosstalk measurements to drive its respacing. Thus, there is often scope for further crosstalk minimization even in routings produced by crosstalk-aware routers. We present a gridless postprocessing algorithm to minimize the peak crosstalk in the nets in a channel by perturbing their horizontal segments. Crosstalk-driven postprocessing algorithms for channels have also been presented in [2, 4].

In [2], the spacing between tightly coupled wires is increased by modeling crosstalk violations between nets as repulsive forces between the horizontal segments of those nets. This is formulated as an Iterative Parameterized Linear Program (IPLP) to respace the wires. However, as mentioned by the authors, this formulation cannot directly handle the coupling between the vertical segments of the wires. (Instead, the vertical coupled lengths are incorporated into the objective function.) Consider the example in Figure 1 having crosstalk violations in the two nets initially. As a result, the IPLP increases the separation between the horizontal segments of these nets. However, the decrease in the coupling between their horizontal segments may be offset by the increased coupling between them due to the increase in their vertical coupled length on the left, causing their final crosstalk to exceed their initial crosstalk. In contrast, our formulation keeps track of the exact crosstalk in each net (including that contributed by vertical segments). Furthermore, we guarantee that the crosstalk characteristics of the nets in the channel never deteriorate as a result of our perturbations.

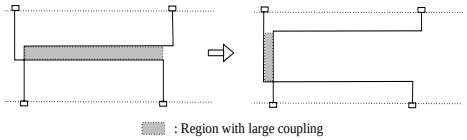

Figure 1: IPLP-based respacing may not always work.

The algorithm presented in [4] attempts to minimize the crosstalk in the nets in a channel by permuting its tracks. Therefore, it is restricted to gridded channels. It formulates the problem as an Integer Linear Program (ILP), with integer variables for each track. Consequently, its running time increases dramatically as the number of tracks in the channel is increased. It is unable to handle the shielding effect between two horizontal wires because of other wires routed between them. As our experiments demonstrate, the crosstalk characteristics of even the optimal track permutations produced by [4] can be improved substantially using our approach.

Even though current fabrication processes can place a wire with an accuracy of 0.02μ, the wire pitch is usually at least 1.2μ ([6]). This justifies the use of gridless routing models. Grid-based algorithms are unable to place the wires optimally due to grid snapping. However, the non-linearity of objective functions involving crosstalk makes it difficult to develop efficient gridless algorithms for their optimization ([2]). In this paper, we show that although these functions are complicated, they are piecewise smooth and "well-behaved", allowing us to converge to the optimal wire locations at each step without having to solve any non-linear equations.

As in most prior works on crosstalk minimization, we assume that the coupling capacitance between two segments with coupled length l, separation d and unit coupling C is given by Cl/d. However, a capacitance model with coupling given by $Cl/d^{1.34}$ was presented in [8]. We note that the theorems presented in our paper are also valid under the model of [8]. Furthermore, our theory is also applicable to more general routing models such as switchboxes and area routing. It can be generalized to determine the optimum location over the entire channel width for each critical trunk. Further details on this paper can be found in [9, Ch. 3].

2 Problem Formulation

The crosstalk between two nets depends on the coupling capacitance between them, other load capacitances, driver strengths, signal voltages, and the temporal correlation between the signals. However, at the layout stage, the only controllable parameter among these is the coupling capacitance. Thus, the total crosstalk in a net can be assumed proportional to the sum of the capacitive couplings

*This research was performed while the authors were with the Department of Computer Science at the University of Illinois at Urbana–Champaign, supported in part by the National Science Foundation under grant MIP-9612184.

DAC 99, New Orleans, Louisiana

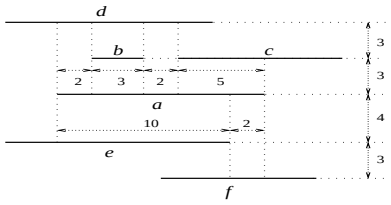

Figure 2: Computing the crosstalk in trunk a

of each segment of the net with its neighbors. Without loss of generality, let us set the constant of proportionality to one.

Let $N_1, N_2, \ldots, N_n$ be n nets routed in a channel. The routing of a net consists of horizontal segments, called *trunks*, connected by vertical segments. Let the routing of net N_i ($i = 1, \ldots, n$) consist of t_i trunks $T_1^{(i)}, \ldots, T_{t_i}^{(i)}$, connected by the v_i vertical segments $V_1^{(i)}, \ldots, V_{v_i}^{(i)}$. Let $\tau_k^{(i,j)}$ ($\xi_k^{(i,j)}$) denote the crosstalk contribution of the trunk T_k (vertical segment V_k) to the crosstalk in trunk $T_j^{(i)}$ (vertical segment $V_j^{(i)}$) of net N_i. If $C_k^{(i,j)}$, $l_k^{(i,j)}$ and $d_k^{(i,j)}$ represent the unit coupling between the two relevant nets, the coupled length between the two segments and the distance separating the segments respectively, then $\tau_k^{(i,j)}$ (or $\xi_k^{(i,j)}$) equals $C_k^{(i,j)} l_k^{(i,j)} / d_k^{(i,j)}$. Observe that although $C_k^{(i,j)}$ is largely determined by the technology, it can also be used to account for signal correlations between different nets. In particular, it is zero between two segments belonging to the same net. Also, if some portion of segment S_k is not visible from segment S_j because of some other segment $S_{k'}$ lying between S_k and S_j, then this hidden portion of S_k does not contribute to the crosstalk in S_j because of the shielding effect of $S_{k'}$. In the example of Figure 2, the portion of the trunk d that lies above trunk b does not contribute to the crosstalk in trunk a. Thus, the coupled length between the two segments can arise from non-contiguous sub-segments, as occurs between trunks a and d in our example. The total crosstalk χ_i in net N_i ($i = 1, \ldots, n$) is given by $\sum_{j=1}^{t_i} \sum_k \tau_k^{(i,j)} + \sum_{j=1}^{v_i} \sum_k \xi_k^{(i,j)}$. For our example, let $C_k^{(i,j)}$ equal 1 for each pair of nets, and the numbers in the figure represent the coupled lengths and separations between the segments. Then, the crosstalk in trunk a due to its upper and lower neighbors is $(\frac{2+2}{6} + \frac{3}{3} + \frac{5}{3})$ and $(\frac{10}{4} + \frac{2}{7})$ respectively, yielding a total of 6.119 units. The total crosstalk in the net to which trunk a belongs is computed by summing up the crosstalks in each of its trunks and vertical segments similarly.

In general, the maximum crosstalk tolerable by a net is usually fixed by the designer. We assume that the maximum tolerable crosstalk in net N_i is represented by the constant B_i. The difference between the maximum tolerable crosstalk B_i and the actual crosstalk χ_i (i.e. $B_i - \chi_i$) is called the *slack* (say, σ_i) in net N_i. Thus, our problem can be stated as: *maximize* $min_{i=1}^{n}\{\sigma_i\}$. The "min" term makes this problem rather difficult. We approach it by *perturbing* the trunks of the various nets so as to maximize the smallest of the σ_i's. Perturbing a trunk involves displacing it *vertically* while keeping the relative vertical ordering between trunks invariant. As we perturb the trunks, we vary the $d_k^{(i,j)}$ terms in the $\tau_k^{(i,j)}$'s (and, consequently, the $l_k^{(i,j)}$ terms in the $\xi_k^{(i,j)}$'s).

3 Trunk Perturbation and the Variation of Net Slacks

In this section, we first present our key theoretical result showing that the perturbation range of a trunk can be efficiently divided into subintervals within which the variation of the slack is "well-behaved". This implies that we can efficiently determine the optimum location for the trunk *without having to solve any non-linear equations*, even though the variation of its slack within its perturbation range is not smooth (and is a polynomial of very high degree even within its smooth segments). Then, in Section 3.2, we study the effect upon neighboring nets when we perturb a trunk belonging to a critical net. The results of this section allow us to structure our perturbation algorithm in a way that guarantees that no non-critical net becomes critical during the perturbation. Finally, in Section 3.3, we show that the perturbation of trunks belonging to *neighbors* of critical nets can be used to further increase the critical slacks (beyond what is possible by trunk perturbations in only the critical nets). Although Sections 3 and 4 have been presented in terms of the net slacks, we would like to point out that they can also be equivalently phrased in terms of the net crosstalks.

The range over which a trunk $T_j^{(i)}$ can be perturbed is bounded by the closest neighboring trunks above and below it. This range can be divided into *Basic Perturbation Intervals* (BPIs) whose end-points are the trunk end-points lying within the perturbation range (to the left or right of $T_j^{(i)}$) that are visible from either of the two vertical lines placed at the two ends of $T_j^{(i)}$. Thus, for the example of Figure 3, the perturbation range for trunk $T_j^{(i)}$ is bounded by T_2 and T_5 and divided into three BPIs due to T_3 and T_4. Observe that the BPIs involved in a particular perturbation can be determined a priori by examining the segments visible from the perturbation range. Furthermore, the total number of trunks provides a (usually conservative) upper bound on the number of BPIs.

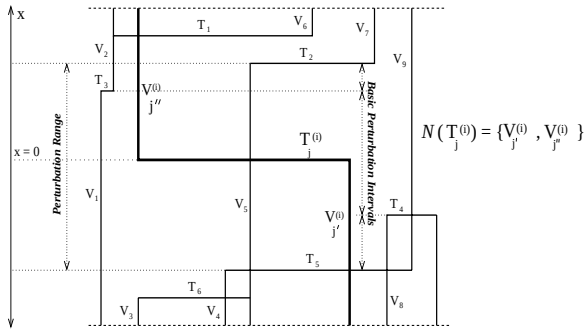

Figure 3: Perturbation range and BPIs for the trunk $T_j^{(i)}$

3.1 Net Slack for a Perturbed Trunk

Theorem 1 (Unimodality Theorem) *When a trunk $T_j^{(i)}$ belonging to net N_i is perturbed, the variation of the slack σ_i in N_i with the displacement of $T_j^{(i)}$ is continuous over the entire perturbation range, and is either monotonic or strictly unimodal (with a unique maximum) within any BPI of the perturbation range.*

Informally, this theorem states that the variation of the slack of a net whose trunk is being perturbed is of the form depicted in Figure 4. It provides us with a powerful tool to determine the maximum of σ_i over the perturbation range without having to solve any non-linear equations. Given a BPI (or a sub-interval of a BPI) $\mathcal{I}$ within the perturbation range, we can easily determine whether the variation of σ_i within $\mathcal{I}$ is monotonic or unimodal, by comparing the signs of its slope at the end-points of $\mathcal{I}$. If the slope has the same sign at both end-points, σ_i attains its maximum over $\mathcal{I}$ at one of the two end-points. If the signs are different, we use binary or golden section search. In this case, this search will indeed converge to the optimum since we are guaranteed that there is a *unique* local maximum of σ_i within $\mathcal{I}$. Finally, we compare the optima obtained for each BPI to determine the optimum over the entire perturbation range. Observe that both the determination of the sign of the slope

of σ_i and the search process involve the computation of $\sigma_i(x)$ at some *specific value of* x, as opposed to the solution of some non-linear equation with a *variable* x.

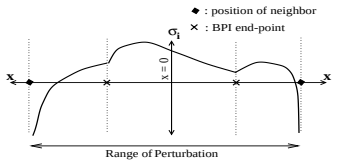

Figure 4: Variation of the net slack σ_i over the perturbation range

3.2 Trunk Perturbation and Neighboring Slacks

The Unimodality Theorem can be used to perturb the trunks belonging to a net in order to improve its slack. However, we would like to guarantee that these perturbations do not cause the crosstalk characteristics of other nets to deteriorate to an extent that they become more critical than the current net. In other words, we would like to perturb trunk $T_j^{(i)}$ (of net N_i) to a location that maximizes the slack σ_i in N_i subject to the constraint that the slacks in nets influenced by the perturbation of $T_j^{(i)}$ do not get any worse than the new value of σ_i. Let $\mathcal{N}(T_j^{(i)})$ denote the set of all the vertical segments of N_i that are adjacent to the trunk $T_j^{(i)}$. (See Figure 3). Then, if some segment of net $N_{i'}$ is visible from some segment in $\{T_j^{(i)}\} \cup \mathcal{N}(T_j^{(i)})$, we must ensure that $\sigma_{i'} - \sigma_i \geq 0$.

Although $\sigma_{i'}$ and σ_i are individually well-behaved in the sense of Theorem 1, the difference of two such unimodal functions is not well-behaved in general (due to terms of the form $\alpha/(\beta \pm x)$ appearing with both positive and negative polarities). However, in the case of the difference of slacks in two neighboring nets, we can make a stronger claim, stated as the following theorem.

Theorem 2 *The variation of the difference of the slacks in nets N_i and $N_{i'}$ when some trunk $T_j^{(i)}$ belonging to N_i is perturbed is continuous over the entire perturbation range, and is either monotonic or strictly unimodal within any BPI of the perturbation range.*

As a corollary, note that the equation $\sigma_i - \sigma_{i'} = 0$ has at most two roots in any BPI $\mathcal{I}$. Therefore, there are at most two disjoint sub-intervals of $\mathcal{I}$ over which $\sigma_{i'} \geq \sigma_i$. These sub-intervals can be determined easily by binary search using the sign of $\sigma_i - \sigma_{i'}$ and its slope. We repeat this process for all the neighboring nets of $\{T_j^{(i)}\} \cup \mathcal{N}(T_j^{(i)})$, and then focus on the intersection of all these sub-interval sets. For any neighboring net $N_{i'}$, $\sigma_{i'}$ is no smaller than σ_i at any point within this intersection. Furthermore, the Unimodality Theorem tells us that the variation of σ_i over any sub-interval of this intersection is also either monotonic or strictly unimodal with a maximum. This enables us to efficiently converge to the maximum of σ_i over $\mathcal{I}$ while guaranteeing that the slacks of neighboring nets do not get any worse than the new value of σ_i. We finally select the best of these BPI-specific maxima over all the BPIs in the perturbation range.

3.3 Perturbation of a Neighboring Trunk

Another perturbation-based method to decrease the crosstalk in any particular trunk $T_j^{(i)}$ is to perturb its neighboring trunks. The perturbation of neighboring trunks can yield further improvement in the crosstalk in a net *even after* its own trunks have been perturbed to their best positions. Consider the example in Figure 5. Let trunk T_1 belong to the most critical net, and T_3 to the least critical net. Let T_1 and T_2 be in their locally optimal positions. Thus, Theorem 2 cannot help decrease the crosstalk in T_1 any further (since moving T_3 away from T_1 will increase the crosstalk in T_2, a trunk that is more critical than T_3). However, since our primary objective is the minimization of the *peak* crosstalk, we should allow such a perturbation, provided T_2 does not become more critical than T_1.

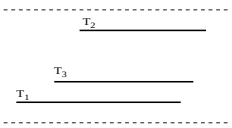

Figure 5: Perturbing the neighbor (T_3) of a critical trunk (T_1)

While perturbing a neighboring trunk $T_{j'}^{(i')}$ in order to decrease the crosstalk in $T_j^{(i)}$, we must ensure that the slack σ_i in the candidate net N_i does not exceed (i) the slack $\sigma_{i'}$ in the net $N_{i'}$ whose trunk is being perturbed, and (ii) the slack $\sigma_{i''}$ in any net $N_{i''}$ that has some segment visible either from the neighboring trunk being perturbed, or from one of its adjacent vertical segments. If we can enforce these two conditions, we will ensure that the improvement in the slack in the candidate net is not negated by excessive deterioration in the slack in some other net affected by the perturbation. As before, we can then determine the subinterval(s) of each BPI in which both our conditions are satisfied, and then determine the optimum location of $T_{j'}^{(i')}$ (that maximizes σ_i) within that region. This would allow us to compare the optima for each BPI to select their maximum, and then perturb $T_{j'}^{(i')}$ to that location.

From Theorem 2, the variation of the difference between the slacks in the candidate net N_i and the neighboring net $N_{i'}$ whose trunk is being perturbed is either monotonic or strictly unimodal over each BPI in the perturbation range. The relation between the slacks in N_i and a neighboring net $N_{i''}$ of the trunk $T_{j'}^{(i')}$ being perturbed is more complicated. However, in most cases, each of N_i and $N_{i''}$ has at most one trunk visible from $T_{j'}^{(i')}$. Therefore, $\sigma_{i''} - \sigma_i$ has at most two terms of the form $\pm C_{k'}^{(i',j')} \, l_{k'}^{(i',j')}/(d_{k'}^{(i',j')} \pm x)$. As a consequence, $\sigma_{i''} - \sigma_i = 0$ reduces to a cubic equation. We can then use the closed-form Cardan formulas ([5, sec. 573-577]) to determine its roots analytically and thus directly determine the sub-intervals of our BPI over which $\sigma_{i''} \geq \sigma_i$.

In the remaining cases, we must resort to search using the sign of $\sigma_{i''} - \sigma_i$. However, we can simplify the search by using "speculative perturbation" as follows. In practice, the variation of $\sigma_{i''} - \sigma_i$ is either unimodal or monotonic over most BPIs. Assume that it is so for all BPIs, use binary search to determine the safe sub-intervals accordingly, and *then* check whether $\sigma_{i''} - \sigma_i \geq 0$ (for each neighboring net $N_{i''}$) at the optimum location. If it is negative for any $N_{i''}$, ignore this optimum. This procedure does not introduce any error into our algorithm; in the worst case, we merely miss out on some perturbations that could have improved the crosstalk characteristics of the channel yet further.

4 The Trunk Perturbation Algorithm

In this section, we use the theoretical results of Section 3 to construct a simple trunk perturbation algorithm (TRUPER) to maximize the minimum of the slacks (or to minimize the maximum of the crosstalks) in the nets in a channel. Our approach is iterative, terminating when an an iteration yields no lexicographic improvement in the crosstalk characteristics of the nets in the channel. In other words, if $(\sigma_{i_1}, \sigma_{i_2}, \ldots, \sigma_{i_n})$ is a list of the slacks in the nets in the channel at the beginning of an iteration arranged in increasing order, and $(\sigma_{i'_1}, \sigma_{i'_2}, \ldots, \sigma_{i'_n})$ is the corresponding list at the end of the iteration arranged in increasing order, our iterations terminate when $\sigma_{i_j} = \sigma_{i'_j}$ for every $j = 1, 2, \ldots, n$. Furthermore,

we guarantee that no iteration causes the crosstalk characteristics of the nets in the channel to deteriorate in a lexicographic sense.

We start each iteration with all the trunks of all the nets being in an "unlocked" state. A trunk is perturbed only if it is unlocked. Within an iteration, we process the nets in increasing order of slack, starting with the most critical net. Given a net, we process its trunks in decreasing order of crosstalk contribution; thus, the most expensive trunks are processed first. While processing any trunk $T_j^{(i)}$ (of net N_i), we first perturb $T_j^{(i)}$ (if it is unlocked) to its optimal position. Next, we perturb each of the currently unlocked trunks visible from it to its position that maximizes the slack in N_i. This is followed by the perturbation of the currently unlocked trunks that determine the end-points of the BPIs for $T_j^{(i)}$. Finally, we lock the current trunk $T_j^{(i)}$ and all its neighboring trunks. (See Figure 6.)

```
repeat
  Unlock all trunks.
  for each net N_i, taken in order of increasing slack σ_i,
    for each trunk T_j^(i) of N_i taken in order of decreasing χ(T_j^(i)),
      if T_j^(i) is unlocked,
        perturb T_j^(i) to maximize σ_i while obeying constraints.
      for each trunk T_k neighboring T_j^(i)
        if T_k is unlocked,
          perturb T_k to maximize σ_i while obeying constraints.
      Lock T_j^(i) and all its neighboring trunks.
until
  no lexicographic improvement in net slacks (or acceptable min slack)
```

Figure 6: Pseudocode for TRUPER

While perturbing any particular trunk, we first identify the BPIs of the perturbation range. Then, we determine the best location for the trunk over each BPI (and thus over the entire perturbation range). All trunk perturbations obey the constraints described in sections 3.2 and 3.3, ensuring that no neighboring net becomes more critical than the one that is currently being optimized.

If we were to attempt the perturbation of a locked trunk, we would usually be unable to displace it from its current location without worsening the slack in the (more critical) net during whose processing it was locked. Thus, the locking mechanism improves the efficiency of TRUPER by allowing us to often avoid the computations involved in perturbations that would yield no improvement. Observe that a trunk is moved *only if* its perturbation improves the slack in either its own net or some other net with equal or less slack, and does not worsen the slack in any other net with slack less than that of the current net. This ensures that no trunk perturbation causes the crosstalk characteristics of the nets to deteriorate lexicographically. TRUPER can be extended further by allowing trunks to be broken up so that each sub-trunk may be perturbed to its locally optimal location. A natural choice for the break-points of a trunk is the visible end-points of its neighboring trunks.

5 Experimental Results

We tested TRUPER with ten benchmark layouts on a Sparc/20 computer. Among these, GL4.opt and Deutsch.opt are the two channel routings published in [4] that have already been optimized for crosstalk by track permutation, the latter being Deutsch's Difficult Example ([3]). De.GTE.* are the channels of the GTE layout published in [3]. YK1, YK4b and YK5 are from [11], while WL16a and LL18 are from [10] and [7], respectively. For our experiments, we minimized the maximum of the net crosstalks. We assumed $C_k^{(i,j)}$ to be 0.3 fF/μ (a value typical of 0.5μ technologies ([1, Fig. 4.4])), the wire pitch (λ) and manufacturing increment to be 1.2μ and 0.04μ, respectively ([6]), and the track and column widths (α) of the initial configurations to range from 1.5μ to 2.0μ.

Table 1 presents the initial and final peak crosstalks ($\chi_{\max}$) for the layouts for α = 1.5μ, 1.8μ and 2.0μ. The parenthesized figures represent the percentage reduction in the peak crosstalk. We obtained substantial improvement in the peak crosstalk characteristics for all the layouts. As expected, the reductions went up in each case as α increased. The average reductions for the three sets of experiments were 14%, 17.6% and 18.5%, respectively. Most of the reduction in the peak crosstalk occurred in the first one or two iterations. The runtimes for each iteration ranged from a few seconds for the smaller layouts to about 50 minutes for Deutsch.opt.

Benchmark	*Initial* $\chi_{\max}$	*Final* $\chi_{\max}$ *(% improvement)*		
		α=1.5μ	α=1.8μ	α=2.0μ
GL4.opt	8.50	7.38 (13.2)	7.11 (16.4)	7.00 (17.6)
Deutsch.opt	138.73	127.22 (8.3)	124.18 (10.5)	123.33 (11.1)
WL16a	7.94	6.81 (14.2)	6.49 (18.2)	6.38 (19.6)
LL18	34.60	23.92 (30.9)	21.26 (38.6)	20.43 (41.0)
YK1	33.08	29.09 (12.0)	28.10 (15.0)	27.53 (16.8)
YK4b	86.94	75.28 (13.4)	70.35 (19.1)	69.56 (20.0)
YK5	53.12	44.78 (15.7)	42.00 (20.9)	41.67 (21.6)
De.GTE.d1	84.52	71.04 (16.0)	68.37 (19.1)	67.61 (20.0)
De.GTE.d3	67.12	57.26 (14.7)	56.66 (15.6)	56.34 (16.1)
De.GTE.d5	106.47	91.31 (14.2)	87.22 (18.1)	86.61 (18.7)
TOTAL	621.02	534.09 (14.0)	511.74 (17.6)	506.46 (18.5)

Table 1: Reduction in maximum crosstalk (λ = 1.2μ, $\chi_{\max}$ in fF)

6 Conclusions

In this paper, we have presented a novel approach to minimize the peak crosstalk in the nets in a gridless channel. We have shown that although the variation of net slacks when a trunk is perturbed is complicated, the perturbation range can be efficiently divided into subintervals within which it is well-behaved. This allows us to determine the optimum location for the perturbed trunk without having to solve any non-linear equations. Our experiments verify that this approach works very well. In particular, the tremendous gains we obtained on each of the layouts that had already been optimally track permuted for crosstalk suggest a two phase post-processing strategy to optimize peak crosstalk in channels – first perform a track permutation ([4]) using a gridded routing model, and then improve the resulting routing further using TRUPER.

References

[1] Bakoglu, H. B., *Circuits, Interconnections and Packaging for VLSI*, Addison-Wesley Publishing Company, 1990.

[2] Chaudhary, K., A. Onazawa and E. S. Kuh, "A Spacing Algorithm for Performance Enhancement and Crosstalk Reduction", *Proc. Intl. Conf. Computer-Aided Design*, 697–702, 1993.

[3] Deutsch, D. N., "A Dogleg Channel Router", *Proc. Design Automation Conf.*, 425–433, 1976.

[4] Gao, T. and C. L. Liu, "Minimum Crosstalk Channel Routing", *IEEE Trans. Computer-Aided Design* **15** (5), 465–474, 1996.

[5] Hall, H. S. and S. R. Knight, *Higher Algebra*, Macmillan & Co. Ltd., London, 4^{th} ed., 1891. Reprinted, 1960.

[6] Karnik, T., Intel Corp., Hillsboro, OR, Private communication, 1997.

[7] Leong, H. W. and C. L. Liu, "A New Channel Router", *Proc. Design Automation Conf.*, 584–590, 1983.

[8] Sakurai, T. and K. Tamaru, "Simple Formulas for Two- and Three-Dimensional Capacitances", *IEEE Trans. Electron Devices* **ED-30** (2), 183–185, 1983.

[9] Saxena, P., *The Retiming and Routing of VLSI Circuits*, Ph.D. Thesis, Tech. Rep. UIUCDCS-R-98-2059, Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign, 1998.

[10] Wong, D. F. and C. L. Liu, "Compacted Channel Routing with Via Placement Restriction", *Integration* **4** (4), 267–307, 1986.

[11] Yoshimura, T. and E. S. Kuh, "Efficient Algorithms for Channel Routing", *IEEE Trans. Computer-Aided Design* **CAD-1** (1), 25–35, 1982.