Optimal Spacing for Managing Cross-talk Induced Noise and Delay in Resistive VLSI Interconnections

Paul B. Morton
Wayne Dai

UCSC-CRL-00-06
April 3, 2000

Jack Baskin School of Engineering
University of California, Santa Cruz
Santa Cruz, CA 95064 USA

ABSTRACT

In this paper we present an efficient algorithm for determining the optimal spacing needed to eliminate cross-talk induced noise violations and cross-talk induced delay violations on nets with nonnegligible interconnect resistance. Since cross-talk violations cannot be accurately detected until very late in the routing process, we base our formulation on the topological representation of the detailed Manhattan routing in order to be able to easily identify and quantify uncommitted routing resources available to a net very late in the routing process. Our experimental results demonstrate that the performance of our algorithm is best characterized as $O(n\,log(n))$ in the number of spacing variables.

Keywords: cross talk, constrained routing, Lagrange relaxation, wire spacing, noise margin, topological routing

1 Introduction

One of the fundamental advantages of digital systems are their ability to reject noise through self restoring logic, that is, the output signal from a logic stage is closer to the ideal logic levels than the input signal. This is the main reason why the vast majority of electronic computing systems are digital, and not analog.

Integrated circuit technology created a way to inexpensively mass produce very reliable and sophisticated digital electronic systems. Integrated circuit technology also brought with it the ability to make continuous and predictable incremental improvements in component density, speed and power consumption. This is accomplished by following a set of scaling rules which systematically reduces feature sizes and power supply levels while giving a high level of assurance that the shrunken devices will still operate correctly. Further density improvements are created through the use of novel gate designs, such as precharged logic. These techniques to improve density, speed, and power consumption also systematically reduce the noise rejection ability of the integrated circuit technology while simultaneously aggravating the mechanisms responsible for cross-talk induced noise and cross-talk induced delay.

There has been substantial progress in the analysis of cross-talk induced noise and delay in resistive VLSI interconnections [1, 2]. This allows us to efficiently and accurately identify nets, in a detailed routing, which have cross-talk problems.

There has been some progress in the area of cross-talk noise management on nets with negligible interconnect resistance (see for example [3, 4, 5, 6, 7, 8]). However, as we know, assuming that nets have negligible interconnect resistance is no longer a valid assumption in the era of deep submicron VLSI design. As depicted in Fig. 1, we see that when a noise signal has been induced on a victim net, a larger noise signal is seen at a victim net's sink pin when the "point of injection" is moved closer to the sink pin and away from the victim net's source pin. This is due to resistive shielding effects. Specifically, as the point of injection moves farther from the source pin, the increased resistance makes it more difficult for the source gate to "absorb" the injected noise pulse. From this we see that the victim net's interconnect resistance plays an important role in the magnitude of the cross-talk noise signals seen at its sink pins, and therefor must be accounted for in any cross-talk noise management strategy.

There has been very little progress for cross-talk induced noise and delay management on nets with nonnegligible interconnect resistance for use in an area routing ("over the cell" routing) environment. Since it is this environment that characterizes todays state of the art VLSI chip designs, we are left in a position where we can identify the cross-talk problems, but then lack the means to effectively correct them.

One of the main reasons for this imbalance is that in order to optimally redistribute uncommitted routing resources to eliminate the cross-talk problems we need a practical solution to the problem of solving a sequence of fairly large non-linear mathematical programs. Compounding the problem is the fact that we need a detailed knowledge of the interaction between adjacent nets in order to accurately detect the cross-talk problems. Because of this it is very difficult to predict cross-talk problems until very late in the routing process. Using a traditional Manhattan router, we find that it is very difficult to correct the cross-talk problems this late in the routing process primarily because of the inflexible form used to represent the underlying routing.

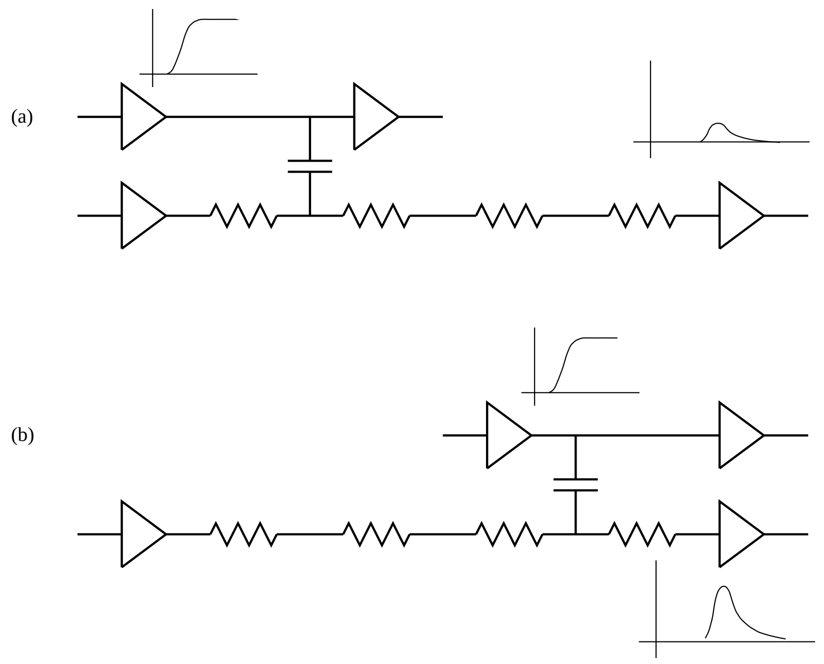

Figure 1: Noise pulse produced when the "point of injection" is (a) close to the victim net's source and far from its sink, and (b) far from the victim net's source and close to its sink.

In [9], the optimal spacing for cross-talk noise management in resistive routing was formulated as a nonlinear programming problem that was solved in $O(n^{3.3})$ time, where n is the size of the victim net. However, the formulation did not include cross-talk induced delay, and its solution has a fairly high computational complexity which makes it impractical for use on large nets.

In this paper we will present an efficient algorithm for determining the optimal spacing which, when sufficient uncommitted routing resources are available, will eliminate cross-talk violation on a given net of a detailed Manhattan routing taking into account the effects of the victim net's interconnect resistance on cross-talk noise. This algorithm will be formulated to eliminate both cross-talk induced noise violation as well as cross-talk induced delay violations and can be used in an area routing environment. In order to more easily identify and quantify the uncommitted routing resources available to a net after a detailed Manhattan routing has been constructed, our formulation will be based on the topological representation of the Manhattan routing. Finally, we will demonstrate that the performance of our algorithm can best be characterized as $O(n\,log(n))$ in the number of spacing variables.

2 Topological Routing and Cross-Talk Management

The main problem in managing cross-talk noise with traditional Manhattan routers is that their underlying representation does not allow for a fine enough, or rich enough set of tools with which to measure and manipulate a detailed routing in order to solve a problem as complex and subtle as the cross-talk problem. To avoid this shortcoming we base our approach on the more powerful and flexible topological representation of a Manhattan routing. In particular, **each layer** of a detailed Manhattan routing can be broken into two components: a topological routing, and a set of branch widths and spacings. Looking at the routing in this form has several key advantages; First, we can easily identify and quantify uncommitted routing resources; Second, concentrating and committing these routing resources can be easily accomplished since this only requires changing the values of the spacing variables, which can be made at virtually no cost; Third, all the information needed to make accurate cross-talk noise estimates can still be easily extracted from the topological routing [10], since it is the routing topology which determines which nets are capacitively coupled.

Using these ideas, and given a design rule correct Manhattan routing, our approach to cross-talk management is as follows:

1. **From a detailed analysis of the Manhattan routing, construct a list of nets with cross-talk induced noise and delay violations.**
2. **For each layer of the Manhattan routing, extract its topological routing.**
3. **In the topological domain, determine a set of spacing values which eliminate the cross-talk violations identified in step 1.**
4. **Construct an improved Manhattan routing from the topological routing and the new spacing values.**

Our approach to cross-talk management can be based on an existing, topological routing system, called SURF [11]. SURF has a rich set of topological routing tools which can be used to easily and efficiently manipulate and measure a topological routing. Once a satisfactory solution has been obtained in the topological domain, SURF can quickly construct the improved detailed Manhattan routing [12].

Ideally, we would like to simultaneously determine the set of spacings for all nets with cross-talk violations. Unfortunately, given the enormous number of nets on todays chip designs, this is clearly an impractical approach. Because of this, we propose a greedy approach. Specifically, we order the nets with cross-talk violations according to the "severity" of their violations. Then a set of spacing values for each net in the list, beginning with the net with the most severe violation, is determined. In order that we have adequate routing resources available to those nets at the end of the list, we need to determine each net's set of spacings such that they eliminate any cross-talk violations while using a minimum amount of routing resources. From this we see that the heart of our new cross-talk management strategy is an optimization problem which is the determination of a victim nets optimal wire spacing under cross-talk constraints. For brevity we will refer to this as "The Optimal Spacing Problem" (OSP).

3 Estimating Cross-talk Induced Noise and Delay

In this paper we represent a net's routing as a tree where all of the information describing a wire segment (including its interaction with adjacent nets) connecting a node, n, and its parent is associated with the node n. That is, all information is stored in the down stream node for the segment.

To estimate the peak cross-talk noise seen at any node of a victim net, we use the Devgan cross-talk noise estimate [13]. The main strengths of Devgan's estimate are its ability to include coupling capacitance, victim net interconnect resistance, and aggressor net rise time in a closed form expression that can be used to quickly and easily analyze networks with complex topologies. This estimate can be computed as

$$V_n \leq \sum_{i \in N} \sum_{m \in CP(i,n)} R_m \sum_{j \in Adj(i)} \phi_{ij}^V C_{ij}^c \dot{U}_j \qquad (1)$$

where V_n is the maximum cross-talk noise voltage induced on node n by all nets adjacent to the victim net. N is the set of all victim net nodes. $CP(i,n)$ is the set of victim net nodes that are common to the path from the root node to node i, and the path from the root node to node n. R_m is the resistance connecting node m to node $Par(m)$, and $Par(m)$ is the parent node of node m. $Adj(i)$ is the set of adjacent nets that are coupled into node i of the victim net. C^c_{ij} is the coupling capacitance between the adjacent net j and the segment connecting node i to node $Par(i)$. $\dot{U}_j$ is the slope of the voltage source driving adjacent net j.

Since a victim net is only sensitive to cross-talk noise during a subinterval of the clock cycle, we have include the factor $\phi^V_{ij} \in \{0,1\}$ to accommodate this behavior. Specifically, for an adjacent net, j, ϕ^V_{ij} is zero when we know that the adjacent net is quiet when the victim net is sensitive to cross-talk noise; ϕ^V_{ij} is one when we can not guarantee that the the adjacent net will be quiet when the victim net is sensitive to cross-talk noise.

To estimate the maximum delay from the source to any node of the victim net, we use the Elmore delay estimate [14]. This estimate can be computed as

$$D_n \leq \sum_{i \in N} \sum_{m \in CP(i,n)} R_m [C^a_i + C^s_i + \sum_{j \in Adj(i)} \phi^D_{ij} C^c_{ij}] \tag{2}$$

where C^a_i is the capacitance associated with the area covered by the wire segment connecting node i to $Par(i)$, and C^s_i is the capacitance associated with any sink pin connected to node i.

The cross-talk induced delay is accounted for by the terms $\phi^D_{ij} C^c_{ij}$. Since the effective coupling capacitance between the victim net and an adjacent net, j, is dependent on the signal transitions occurring on both nets, we have included the factor $\phi^D_{ij} \in \{0,1,2\}$ to accommodate this behavior. Specifically, when we know that the victim net and the adjacent net make simultaneous transitions in the same direction, ϕ^D_{ij} is zero; when we know that the adjacent net will be quiet during all victim net transitions, ϕ^D_{ij} is one; when we can not grantee either of these two cases, then we assume that the victim and the adjacent net make simultaneous transitions in opposite directions, which can be accounted for by setting ϕ^D_{ij} to two.

To account for the effects of the victim net's driver resistance, R_S, on both cross-talk induced noise and cross-talk induced delay, we define $R_{root} = R_S$, $C^a_{root} = 0$, $C^s_{root} = 0$, and $Adj(root) = \emptyset$. Additionally, it should be noted that we have chosen to use the more conservative form of the Elmore delay which lumps all the capacitance associated with a wire segment at the far end of the segment. This has be done strictly in the interest of clarity and brevity. All the techniques presented in this paper can easily be extended to the use of the less conservative π model representation of a wire segment.

4 Formulating the Optimal Spacing Problem

We associate with each node of the victim net's routing tree a spacing variable S_i. This spacing, as shown in Fig 2, represents the minimum distance allowed between the wire segment, connecting node i to $Par(i)$, and any adjacent wire segments. For the purpose of computational convenience we have also included the required minimum spacing, S_{min}, between adjacent wire segments as part of S_i. From this we see that

$$C^c_{ij} \leq C^c_T \frac{L_{ij}}{S_i} \quad \forall\, j \in Adj(i) \tag{3}$$

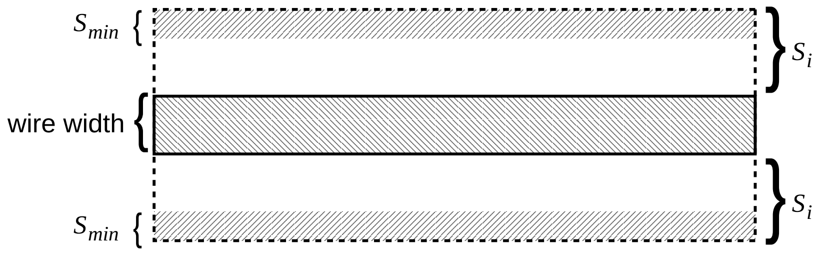

Figure 2: Wire spacing model.

where L_{ij} is the estimated length of the adjacency between adjacent net j and the wire segment connecting node i to node $Par(i)$, and C_T^c is a proportionality constant determined from the technology. Substituting (3) into (1), and (2) we have

$$V_n \le \sum_{i \in N} \sum_{m \in CP(i,n)} R_m \sum_{j \in Adj(i)} \phi_{ij}^V C_T^c \frac{L_{ij}}{S_i} \dot{U}_j \tag{4}$$

$$D_n \le \sum_{i \in N} \sum_{m \in CP(i,n)} R_m [C_i^a + C_i^s + \sum_{j \in Adj(i)} \phi_{ij}^D C_T^c \frac{L_{ij}}{S_i}] \tag{5}$$

The routing resources consumed by the extra spacing allocated to a net can be measured as

$$A = \sum_{n \in N} 2\, L_n (S_n - S_{min}) \tag{6}$$

where L_n is the estimated length of the wire segment connecting node n to $Par(n)$.

Using (4), (5), and (6) we can formulate the OSP problem as the following constrained minimization problem:

$$\boldsymbol{\min_{\overline{S}} \{ \sum_{n \in N} L_n\, S_n \}} \tag{7}$$

subject to

$$\boldsymbol{S_n \ge S_{min}} \quad \forall\, \boldsymbol{n \in N} \tag{8}$$
$$\boldsymbol{\overline{f}(\overline{S}) \le \overline{B}} \tag{9}$$
$$\boldsymbol{V_n \le M_n} \quad \forall\, \boldsymbol{n \in Pins(N)} \tag{10}$$
$$\boldsymbol{D_n \le Q_n} \quad \forall\, \boldsymbol{n \in Pins(N)} \tag{11}$$

where $\overline{S}$ is a vector of all spacing variables. (8) are lower bounds imposed by the technology. (9) are routing resource constraints, or the "upper bounds" imposed by the routing topology. (10) are the cross-talk noise constraints. (11) are the delay constraints. $Pins(N)$ is the subset of victim net nodes connected to sink pins. M_n is the noise margin for node n of the victim net. Q_n is the maximum allowable delay from the source of the victim net to node n of the net.

In general, the upper bound on each S_n is dependent on other spacing variables. In order to accommodate this behavior, the upper bound on each S_n is written, in (9), as a vector valued function, $\overline{f}$, of all the spacing variables. In general, $\overline{B}$ is a vector of positive constants and the function $\overline{f}$ is a linear function with positive coefficients, as we will see in section 5. The exact form of $\overline{f}$ is determined by the routing topology and the value of the coefficients are determined from the state of the routing just prior to the optimization process.

After solving this optimization problem we can use (12) to determine the amount of extra routing resources, in the form of extra spacing, to commit to each side of every wire segment in the victim net.

$$S_n^e = S_n - S_{min} \quad \forall\, n \in N \tag{12}$$

5 Formulating the Upper Bound Constraints

The key advantage to working with a topological routing is the ability to identify, quantify, and concentrate the uncommitted routing resources available to a net once the design has been routed. In essence the topological representation of a routing gives us the ability to easily "look through" a net's immediately adjacent routing and take advantage of routing resources that would otherwise be very difficult to utilize in a traditional Manhattan routing representation. Specifically, in a topological routing we can take advantage of Maley's routability theorem [15] to determine the set of upper bound constraints represented by (9). Maley's routability theorem states, in part, that a net's topological routing is routable if, and only if, the flow does not exceed the capacity of each cut the net encounters (crosses or intersects). A cut is defined as the shortest straight line between two features that are visible to each other. A feature is any object through which a branch of the topological routing cannot be routed, excluding other branches. Fig. 3 illustrates a cut between two terminals of a topological routing. Note that a terminal can represent a pin, a via contact, branch point, or a pad.

The capacity of a cut, $capacity(c)$, as illustrated in Fig. 3, is a measure of the amount of routing resources available across the cut. The flow of a cut, $flow(c)$, is a measure of the amount of routing resources needed to route all the branches that need to cross the cut, as illustrated in Fig. 3. The flow must include the width of each branch, as well as the spacing needed to separate each pair of adjacent branches, and any spacing needed to separate the branches from the two features which define the end points of the cut, as well as any space needed to accommodate the geometry of the terminals.

From these definitions we see that (9) can be written as

$$flow(c) \leq capacity(c) \quad \forall\, c \in C \tag{13}$$

where C is the set of all cuts that the victim net encounters.

Since the cuts that impose the tightest bounds on $\overline{S}$ are most likely going to be the shortest cuts that the net encounters, and the shortest cuts, in turn, have a low probability of involving more than one branch of the net, we see that the vast majority of the upper bound constraints will be in the form of fixed budget constraints.

$$S_n \leq S_{n-budget} \tag{14}$$

Further, since only a small number of cuts will involve more than one spacing variable, we can, without sacrificing too much of the OSP's feasible region, significantly simplify the solution of the OSP by restricting these linear constraint equations to a set of fixed budget constraints. In particular, if we have an upper bound constraint of the form

$$S_1 + S_2 \leq S_{cut} \tag{15}$$

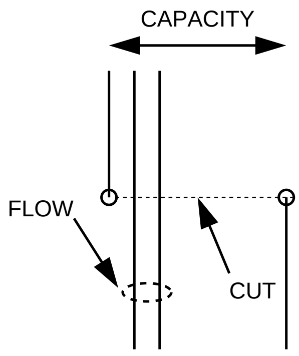

Figure 3: The flow and capacity of a cut between two terminals of a topological routing.

then we can convert this to the following set of fixed budget constraints

$$S_1 \leq S_{1-budget} = \frac{S_{cut}}{2} \tag{16}$$

$$S_2 \leq S_{2-budget} = \frac{S_{cut}}{2} \tag{17}$$

Using these ideas, we see that (9) can be reduced to

$$S_n \leq S_{n-budget} \quad \forall\, n \in N \tag{18}$$

6 Solving the Optimal Spacing Problem

Because (10) and (11) are nonlinear functions of S_n, the OSP forms a nonlinear programming problem. However, (10) and (11) are convex over the region defined by $S_n \geq S_{min}$, and thus the OSP forms a convex programming problem which can be solved using Lagrange relaxation [16, 17]. Specifically, we associate with each sink pin, p, a pair of Lagrange multipliers, λ_p and γ_p, which are used to relax that pin's noise and delay constraints, giving us the following Lagrange subproblem (LSP):

$$\min_{\overline{S}}\{\sum_{n\in N} L_n\, S_n + \sum_{p\in Pins(N)} \lambda_p(V_p - M_p) + \sum_{p\in Pins(N)} \gamma_p(D_p - Q_p)\} \tag{19}$$

subject to

$$S_n \geq S_{min} \qquad \forall\, n \in N \tag{20}$$

$$S_n \leq S_{n-budget} \qquad \forall\, n \in N \tag{21}$$

Using the subgradient optimization technique outline in Fig 4 we can solve the Lagrange dual problem (LDP) giving us an optimal set of spacings which also solve the OSP. To solve the LSP, we take advantage of the fact that (19) is separable in $\overline{S}$ and can be written as

$$\min_{\overline{S}}\{\sum_{q\in N} g_q(S_q)\} \tag{22}$$

where

$$g_q(S_q) = K_{1q}S_q + \frac{K_{2q}}{S_q} + K_{3q} \tag{23}$$

```
1. $\lambda_p \leftarrow 0 \; \forall \; p \in Pins(N)$
2. $\gamma_p \leftarrow 0 \; \forall \; p \in Pins(N)$
3. $i \leftarrow 1$
4. Repeat
5.      Solve LSP
6.      $\lambda_p \leftarrow max(0, \lambda_p + \theta_i(V_p - M_p)) \; \forall \; p \in Pins(N)$
7.      $\gamma_p \leftarrow max(0, \gamma_p + \theta_i(D_p - Q_p)) \; \forall \; p \in Pins(N)$
8.      Update $\theta_i$ such that $\lim_{i \to \infty} \theta_i = 0$
          and $\sum_{j=1}^{i} \theta_j \to \infty$
9.      $i \leftarrow i + 1$
10. Until $\Delta\lambda_p$ and $\Delta\gamma_p \leq$ error bounds $\forall \; p \in Pins(N)$
```

Figure 4: Subgradient optimization algorithm for solving the Lagrange dual problem.

From the Kuhn-Tucker optimality conditions [18] we find that the LSP can be solved as

$$S_q = min(S_{q-budget}, max(S_{min}, \sqrt{K_{2q}/K_{1q}})) \tag{24}$$

where

$$K_{1q} = L_q \tag{25}$$

To determine K_{2q} we collect all terms of (19) which contain $1/S_q$. This gives us

$$\begin{aligned} K_{2q} &= \sum_{p \in Pins(N)} \lambda_p \sum_{m \in CP(q,p)} R_m \sum_{j \in Adj(q)} \phi_{qj}^V C_T^c L_{qj} \dot{U}_j \\ &+ \sum_{p \in Pins(N)} \gamma_p \sum_{m \in CP(q,p)} R_m \sum_{j \in Adj(q)} \phi_{qj}^D C_T^c L_{qj} \end{aligned} \tag{26}$$

Rearranging the order of summation we have

$$\begin{aligned} K_{2q} &= \sum_{m \in Ans(q)} R_m \sum_{p \in DesPins(m)} \lambda_p \sum_{j \in Adj(q)} \phi_{qj}^V C_T^c L_{qj} \dot{U}_j \\ &+ \sum_{m \in Ans(q)} R_m \sum_{p \in DesPins(m)} \gamma_p \sum_{j \in Adj(q)} \phi_{qj}^D C_T^c L_{qj} \end{aligned} \tag{27}$$

where $Ans(q)$ is the set of nodes which contains q and all of its ancestors. $DesPins(m)$ is the set of m's descendant nodes that contain sink pins and the node m if it contains a sink pin. This can be written as

$$K_{2q} = R_q^V \sum_{j \in Adj(q)} \phi_{qj}^V C_T^c L_{qj} \dot{U}_j + R_q^D \sum_{j \in Adj(q)} \phi_{qj}^D C_T^c L_{qj} \tag{28}$$

where R_q^V and R_q^D are the "weighted up stream resistances" [19] seen from node q and are computed as

$$R_q^V = \sum_{j \in Ans(q)} \Lambda_j R_j \tag{29}$$

$$R_q^D = \sum_{j \in Ans(q)} \Gamma_j R_j \tag{30}$$

where the "weighting factors", Λ_j and Γ_j, associated with node j are computed as

$$\Lambda_j = \sum_{p \in DesPins(j)} \lambda_p \tag{31}$$

$$\Gamma_j = \sum_{p \in DesPins(j)} \gamma_p \tag{32}$$

From (31) and (32) we see that the set of weighting factors can be computed in one bottom up traversal of the routing tree. From (29) and (30) we see that the weighted up stream resistances can be computed in a single top down traversal of the routing tree. Further, once the weighted up stream resistances for a node have been computed we can immediately compute the node's spacing using (28), (25), and (24). From these three observations we have the following:

Theorem 1: *The LSP can be solved in $O(q)$ time, where q is the number of wire segments in the routing tree.*

From lines 6 and 7 of Fig. 4, we see that each iteration of the solution of the LDP must compute a new set of Lagrange multipliers. This can be accomplished by two traversals of the routing tree. One bottom up traversal to compute the down stream current and down stream capacitance seen by each node, and one top down traversal to compute the noise and delay seen at each sink pin. From these observations, Theorem 1, and the fact that the number of sink pins is bounded above by the number of segments in the net, we have the following:

Theorem 2: *Each iteration in the solution of the LDP requires $O(q)$ time.*

Finally, we note that we can easily determine if there are no feasible solutions to the OSP by checking to see if (10) and (11) can be satisfied when all spacing variables are set to the maximum spacing allowed by (18). If they cannot be satisfied then we do not have enough uncommitted routing resources available to the net in order to eliminate all of the net's cross-talk violations.

7 Experimental Results

To evaluate the performance of our optimal spacing algorithm we have tested it on a large set of randomly generated nets. Our results have been generated on a Sun Enterprise 450 ($300\,MHz$ Ultra SPARC-II CPU) with $1\,GB$ of memory. The nets were generated based on the following technology parameters: The minimum spacing between wires is $0.33\,\mu m$; The wire resistivity is $0.291\,\Omega/\mu m$; The capacitive coupling between adjacent wires separated by the minimum wire spacing is $0.745\,fF/\mu m$; The area capacitance is $0.745\,fF/\mu m$; The supply voltage is $1.5V$. The rise times of the adjacent nets were randomly selected between $20\,pS$ and $500\,pS$; The source gates output resistance is $100\,\Omega$; The budget for each spacing variable, S_i, is randomly selected between $0.385\,\mu m$ and $3.08\,\mu m$. The noise and delay margins, M_n and Q_n, for each sink pin were randomly selected from the range of feasible noise and delay margins for that sink pin. The lower bounds on the feasible ranges for each sink pin are determined by computing the noise and delay seen at each pin assuming that the maximum allowed spacing is used around each of the net's wire segments. Similarly, the upper bound on the feasible ranges for each sink pin can be determined by assuming that the minimum spacing is used around each segment.

For each of the randomly generated nets, we determined the CPU time needed by our algorithm, coded in C++, to find an optimal set of spacings for the net. These results are shown in Fig. 5. From these results we found that the computational complexity of our algorithm can best be described as $O(n\,log(n))$, as indicated by the line in Fig. 5.

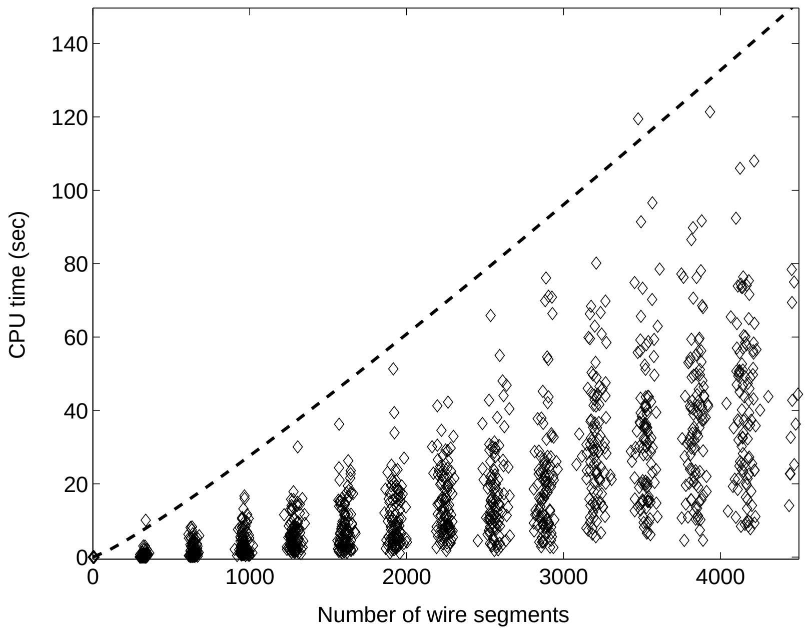

Figure 5: CPU time to compute optimal spacing.

8 Conclusion

In this paper we have presented an efficient algorithm for determining the optimal spacing needed to eliminate cross-talk induced noise violations and cross-talk induced delay violations on nets with nonnegligible interconnect resistance. To estimate the cross-talk induced noise and the cross-talk induced delay we have used the Devgan noise estimate and the Elmore delay estimate, respectively. We have used the topological routing representation of the detailed Manhattan routing to allow us to easily identify and quantify the uncommitted routing resources which are available to a net late in the routing process. Using these estimates and routing resource constraints we formulate a nonlinear programming problem whose solution is the optimal set of spacings that will eliminate the nets cross-talk violations. We solve this nonlinear programming problem using Lagrange relaxation. Our experimental results demonstrate that the performance of our Lagrange relaxation algorithm is best characterized as $O(n\,log(n))$ in the number of spacing variables.

References

[1] K. Shepard, V. Narayanan, and R. Rose, "Harmony: Static Noise Analysis of Deep Submicron Digital Integrated Circuits," *IEEE Trans. Computer-Aided Design*, vol. 18, no. 8, pp. 1132–1150, 1999.

[2] D. Blaauw, A. Devgan, and A. Dharchoudhury, "ICCAD-99 Tutorial on Signal Integrity in High Performance Design," in *Course Material*, 1999.

[3] T. Xue, E. Kuh, and D. Wang, "Post Global Routing Crosstalk Risk Estimation and Reduction," in *Proc. IEEE/ACM Int. Conf. on Computer Aided Design*, pp. 302–309, 1996.

[4] H. Zhou and D. Wong, "Global Routing with Crosstalk Constraints," in *Proc. Design Automation Conf.*, pp. 374–377, 1998.

[5] H. Zhou and D. Wong, "Crosstalk-Constrained Maze Routing Based on Lagrangian Relaxation," in *Proc. IEEE Int. Conf. on Computer Design*, pp. 628–633, 1997.

[6] T. Gao and C. Liu, "Minimum Crosstalk Channel Routing," *IEEE Trans. Computer-Aided Design*, vol. 15, no. 5, pp. 465–474, 1996.

[7] P. Saxena and C. Liu, "Crosstalk Minimization using Wire Perturbations," in *Proc. Design Automation Conf.*, pp. 100–103, 1999.

[8] A. Onozawa, K. Chaudhary, and E. Kuh, "Performance Driven Spacing Algorithms using Attractive and Repulsive Constraints for Submicron LSI's," *IEEE Trans. Computer-Aided Design*, vol. 14, no. 6, pp. 707–719, 1995.

[9] P. Morton and W. Dai, "An Efficient Sequential Quadratic Programming Formulation of Optimal Wire Spacing for Cross-Talk Noise Avoidance Routing," in *Proc. Int. Symposium on Physical Design*, pp. 22–28, 1999.

[10] J. Su and W. Dai, "Post-Route Optimization for Improved Yield Using a Rubber-Band Wiring Model," in *Proc. IEEE/ACM Int. Conf. on Computer Aided Design*, pp. 700–706, 1997.

[11] D. J. Staepelaere, J. Jue, T. Dayan, and W. W.-M. Dai, "Surf: a rubber-band routing system for multichip modules," in *Proc. IEEE Design and Test of Computers*, 1993.

[12] D. J. Staepelaere, "Geometric transformation for a rubber-band sketch," Master's thesis, University of California Santa Cruz, Septemeber 1991.

[13] A. Devgan, "Efficient coupled noise estimation for on-chip interconnects," in *Proc. IEEE/ACM Int. Conf. on Computer Aided Design*, pp. 147–151, 1997.

[14] W. Elmore, "The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers," *Journal of Applied Physics*, vol. 19, no. 1, pp. 55–63, 1948.

[15] F. Maley, *Single-Layer Wire Routing and Compaction.* Cambridge, Mass: The MIT Press, 1989.

[16] M. Fisher, "An Applications Oriented Guide to Lagrangian Relaxation," *Interfaces*, vol. 15, no. 2, pp. 10–21, 1985.

[17] M. Bazaraa, H. Sherali, and C. Shetty, *Nonlinear Programming.* New York: John Wiley and Sons, 1993.

[18] D. Luenberger, *Linear and Nonlinear Programming.* Reading, MA: Addison-Wesely, 1984.

[19] C. Chu and D. Wong, "Greedy Wire-Sizing Is Linear Time," *IEEE Trans. Computer-Aided Design*, vol. 18, no. 4, pp. 398–405, 1999.