Post-Route Gate Sizing for Crosstalk Noise Reduction

Murat R. Becer, David Blaauw*, Ilan Algor, Rajendran Panda, Chanhee Oh,
Vladimir Zolotov and Ibrahim N. Hajj†
Motorola Inc., Univ. of Michigan Ann Arbor*, Univ. of Illinois Urbana-Champaign†

Abstract

Gate sizing is a practical and a feasible crosstalk noise correction technique in the post route design stage, especially for block level sea-of-gates designs. The difficulty in gate sizing for noise reduction is that by increasing a driver size, noise at the driver output is reduced, but noise injected by that driver on other nets is increased. This can create cyclical dependencies between nets in the circuit with noise violations. In this paper, we propose a fast and effective heuristic post-route gate sizing algorithm that uses a graph representation of the noise dependencies between nodes. Our method utilizes gate sizing in both directions and works in linear time as a function of the number of gates. The effectiveness of the algorithm is shown on several high performance designs.

1 Introduction

Crosstalk noise is a critical design and verification issue for large, high-performance designs. This problem has become more significant due to the increased ratio of crosstalk capacitance to total capacitance of a wire and the usage of more aggressive and less noise immune circuit structures, such as dynamic logic.

In noise analysis, the nets on which crosstalk noise is injected by one or more of its neighbors are called the *victim* nets whereas the nets that inject this noise are called the *aggressor* nets. Noise can be divided into two types: *Functional noise* refers to noise that occurs on a victim net which is being held quiet by a driver. Crosstalk noise on such a victim causes a glitch which may propagate to a dynamic node or a latch, changing the circuit state and causing a functional failure [1, 2]. On the other hand, *delay noise* refers to noise that occurs when two capacitively coupled nets switch simultaneously. Depending on the direction of these transitions, the delays on both nets are affected [3, 4]. The focus of this paper is on functional noise but the presented techniques can be used for delay noise as well.

Recent literature proposes a number of crosstalk noise analysis and noise avoidance methods. [5] and [1] propose detailed noise analysis using parasitic extraction and model order reduction. In [6, 7], the authors introduce simple noise metrics for crosstalk amplitude and pulse width in capacitively coupled interconnects. The derived expressions are also used to motivate circuit design techniques, such as transistor sizing and layout techniques to reduce crosstalk. In other recent works, [8] proposes an improved 2π model which is extended by [9] to a 4π model. [9] also uses this model to analyze the effects of several circuit parameters to noise, giving guidelines to the effectiveness of several noise reduction methods.

In this paper, we focus on correcting the identified crosstalk noise problems in the post-route design stage. Noise can be reduced through routing and interconnect optimization (wire spacing, wire widening, controlling coupling length and position) [10, 11], buffer insertion [12, 13] and driver sizing. In the post-route design stage, it is not desirable to use techniques such as wire perturbations and buffer insertion since they would require re-routing and thus increase design time. Re-routing can result in significant changes in net lengths and neighbors of nets which can cause many new noise failures that did not exist initially. After routing is completed, noise failures are therefore more effectively corrected using driver sizing. The flexibility through scalable libraries and existing fill-space, allows one to make incremental changes to the driver sizes without affecting global routing.

Recently, [14, 15, 16, 17] propose transistor sizing methods for crosstalk noise reduction. Reference [14] uses coupling capacitance as the noise metric and optimizes noise, area, delay and power by gate and wire sizing. A gate sizing method to reduce crosstalk induced delay noise is proposed in [15] and is based on a crosstalk noise aware static timing analysis. More recently, in [17], a post-route gate sizing algorithm for crosstalk noise reduction is proposed. However, since this method only utilizes downsizing of aggressor drivers under delay constraints, and not increasing the size of victim drivers, it has limited effectiveness.

In this paper, we therefore propose a new post-route gate sizing algorithm for crosstalk noise reduction. The algorithm increases the size of victim drivers as well as reducing the size of aggressor drivers. The proposed algorithm takes into account both timing and area constraints and treats each net as both an aggressor as well as a victim. This duality is a crit-

Proceedings of the Fourth International Symposium on Quality Electronic Design (ISQED'03)
0-7695-1881-8/03 $17.00 © 2003 IEEE

ical factor in post-route gate sizing and must be accounted for to ensure that new noise violations are not introduced while fixing existing failures. We approach the problem by introducing a noise graph which is constructed based on the static noise analysis of the design. The noise graph represents all critical nets, their significant aggressors, and the dependencies between them. We introduce a sensitivity measure to eliminate weak dependencies from the noise graph to reduce system complexity. We then eliminate cycles from the graph by removing a minimum number of vertices and then sort the resulting acyclical graph topologically. Gate sizing is then performed on the sorted noise graph under delay, area and optimality constraints. The algorithm is guaranteed to converge and has a runtime complexity that is linear with the size of the circuit. Results on three large microprocessor designs are presented to demonstrate the effectiveness of the approach.

The paper is organized as follows. Section 2 outlines a brief qualitative analysis of post-route gate sizing. In Section 3, we explain the noise graph concept and our algorithm in detail including the cycle breaking strategy and sensitivity measure. Results on three high performance microprocessor designs are presented in Section 4. Section 5 contains closing remarks.

2 Gate Sizing for Noise Reduction

If a victim driver is sized up, its effective conductance increases and more effectively holds a net at a steady voltage (v_{dd} or ground). On the other hand, if an aggressor driver is sized down, its effective conductance decreases and as a result, noise induced by the aggressor on a victim net decreases. Figure 1 shows a situation where several nets form a coupled cluster. We investigate the noise pulses at the receiver input and output of net *v* and *a_1* as the driver gate of net *v* is sized up. Figure 2 (b) shows the voltage waveforms at receiver input (falling glitches) and receiver output (rising glitches) of net *v* when the driving gate of net *v* is varied from inv_4 to inv_12. The inverters are from a high performance standard cell library and their transistor widths are proportional to x in inv_x. As can be seen, the noise pulse at the receiver input of net *v* is reduced in terms of both noise peak and noise width, as the driving gate is sized up. Note that, although the noise pulse peak with inv_12 is nearly $450mV (35\%$ of $V_{dd})$, the propagated noise at the receiver output is negligible. The complication of victim-aggressor duality in driver sizing emerges when we consider the voltage waveforms on net *a_1* (Figure 2 (a)). In this figure, the high to low transition is when the net *v* is switching. As can be seen, when the driver of *v* is sized up from inv_4 to inv_12, the transition on net *v* becomes faster, making it a stronger aggressor on net *a_1*. This causes the noise pulse at the receiver input of *a_1* to increase about $50mV$ resulting in the propagated noise pulse to increase by $200mV$. As demonstrated in this example, when a net's driver gate is sized up to reduce noise at that net, it also becomes a stronger aggressor which in turn can induce more noise on other nets. Thus, a driver sizing solution on nets with existing noise failures, may result in new failures on other nets. Post route driver sizing therefore must account for this dependence between nets to ensure convergence of the algorithm.

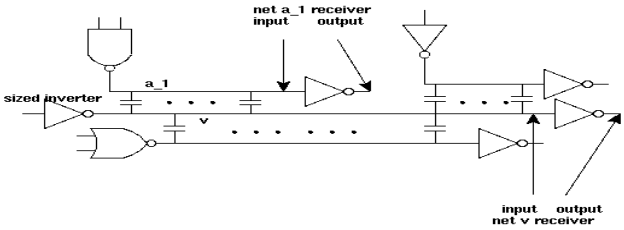

Figure 1. A coupled net cluster

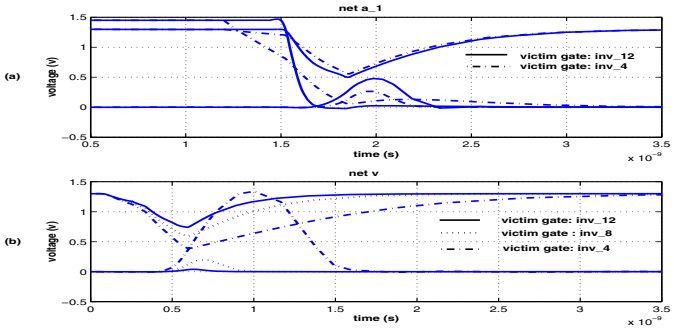

Figure 2. Noise reduction with driver size-up

Other critical complications in the post-route driver sizing problem can be explained as follows:

Area impact: When a driver is sized up, it requires a larger footprint and may cause the legalizor to shift around some of the neighboring instances causing some changes in the routing. Usually, the additional area required to size up a driver gate (i.e. replacing it with a bigger size from the library) is less than the area required to insert a buffer and is less likely to modify the route significantly. Also, the existing fill space around the instances can be utilized.

Timing impact: When a gate size is changed, it effects the timing of paths through this gate. Sizing up a gate speeds up the signal on the net that it drives but it also presents a higher gate capacitance to the previous net, slowing it down. Reverse effects are true if a gate is sized down. Effects on the previous net can be eliminated if the gates in the library are designed in multiple stages, keeping the first stage size the same and reflecting the size differences in the driving stage. During gate sizing, effects on timing are represented as constraints on gate sizes.

System size: The victim-aggressor duality dictates that all interacting nets and their driver gate sizes should be taken into account in a gate sizing algorithm. This makes the problem a constrained multiple goal attainment problem for which the system size ($> 100k$ nets is very common) can be prohibitive. An exact solution is therefore not possible, and

we propose a heuristic solution in this paper.

3 Proposed Gate Sizing Method

In a post-route design stage, detailed information on the topology, neighbors and drivers of all the nets in the design is available. First, we perform an accurate post-route static noise analysis on the design using [1]. Noise analysis identifies the severity of noise on each net through a "slack" value. If the slack of a net is negative, it is failing the noise analysis. The failure criterion used in the paper is the so called 'Noise Rejection Curve' method [1]. Each cell in the standard cell library is characterized with a noise rejection curve which shows the $Height/Width$ boundary at which the cell starts to propagate more than a predefined output noise threshold ($noise\_slack = output\_noise\_threshold - V(receiver\_output)$) [18]. It is also important to avoid unnecessary pessimism in noise analysis which could introduce many false failures. To reduce pessimism, we utilize timing windows and logic correlation information in the noise analysis.

3.1 Noise Graph Representation

As explained in this section, we represent the gate sizing problem using a noise graph. A noise graph $G((V, A), E)$ consists of vertices (V, A) and edges E:

- Vertices: Type V vertices represent nets whose drivers are candidates to be sized up. A type V vertex represents a net which is failing noise analysis or close to failing. In other words nets that have slack less than some predefined positive value will be of type V vertices. Type A vertices represent significant aggressors which have very low noise on them. A significant aggressor is an aggressor which contributes at least 20% of the total noise on a victim net. Very low noise means that net has a slack greater than a predefined positive slack. Type A vertices represent nets whose drivers are candidates to be sized down.

- Edges: A directed edge between vertex a and vertex b exists if net a is a significant aggressor of net b. Note that type A vertices always have an in-degree of 0.

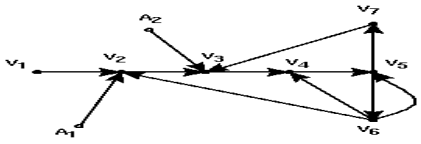

Figure 3. A sample noise graph

Figure 3 shows a simple noise graph. It is a directed graph which contains cycles. The noise graph contains all the failing and critical nets $(V1 - V7)$ as well as their very low noise aggressors $(A1 - A2)$. It also contains the existing significant relations between these nets in the form of edges. In reality, victim-aggressor duality exists for each neighboring net in the design. However, in our noise graph, we incorporate only significant edges, filtering out the insignificant victim-aggressor dependencies which would otherwise increase complexity. Therefore, cycles in the noise graph represent significant victim-aggressor dependencies (in some cases in a more extended sense – $V3 - V4 - V5 - V7 - V3$ cycle). These cycles may lead to oscillating solutions and/or convergence problems. In the simplest case of a two vertex cycle, made up of vertices $V5$ and $V6$, the negative slack can oscillate between the two nets as each one is sized up and neither is fixed. Noise graph also dictates an order in which type V vertices are sized up. For example, if we first size up $V2$ and then $V1$, we might have to come back to $V2$ as it is affected by $V1$. This information is utilized to minimize the complexity of our algorithm.

3.2 Sizing Algorithm

Our algorithm can be summarized as follows. After constructing the noise graph as explained in the previous subsection, we first size down all type A vertices. At this point, if the noise graph is acyclic, we simply size up the type V vertices in topological order. However, in general, the noise graph will contain cycles which may lead to problems as explained above. We address this issue by eliminating the cycles through the removal of some type V vertices from the noise graph. The resulting directed acyclic graph is then topologically sorted and type V vertices are again sized up in the topological order.

Algorithm: Post-route driver sizing
Input: Noise analysis results
Output: Instance cell replace directives
begin
1 Construct a noise graph $G = ((V, A), E)$ based on noise analysis
2 Size_down_type_A_vertices(G)
3 Break_cycles(G)
4 G_s = Topologic_sort(G)
5 **for** each vertex v in G_s
6 Size_up(v)
end

We now explain the algorithm stages in detail. In Step 1, we construct a noise graph based on the noise analysis. During the construction of the noise graph, we apply a sensitivity based pruning method to further eliminate some of the introduced edges. As explained in subsection 3.1, an edge e from vertex u to vertex v represents a significant noise contribution from the net represented by vertex u to the net represented by vertex v. We add a dynamic character to this static edge insertion criterion by introducing a sensitivity notion. As the driver of vertex u is sized up, if the noise change on vertex v is not significant, i.e., $\Delta(noise_v)/\Delta(size_u)$ is very small, then we can conclude

that when vertex u is sized up, this will not increase the noise on vertex v. In other words, the noise dependency from u to v is weak. Edges that represent such weak dependencies are eliminated. In Step 2, we size down all type A vertices as much as possible such that they maintain a sufficient noise slack margin and stay within the timing constraints. By sizing down the significant aggressors up front, the rest of the algorithm is simplified since from this point on only size-up operations will be performed. The constraint on the $noise\_slack$ of type A vertices ensures that no new failures among these nets will be introduced, while trying to fix the existing failing nets.

Algorithm: Size_down_type_A_vertices(G)
Input: G
Output: G with some type A vertex drivers sized down
begin
1 **for** each type A vertex v_a in G
2 Find smallest, same functionality cell in library such that $noise\_slack(v_a)$ and $time\_slack(v_a)$ are within constraints
end

In Step 3, we remove any cycles in the graph by eliminating some vertices. By breaking cycles, we sacrifice some nets (they will not be fixed by driver sizing), but we ensure that there will not be any convergence issues. Our cycle breaking strategy (Break_cycles(G)) ensures that minimum number of type V vertices are removed from the noise graph: Let G be a directed graph $G = (V, E)$ where V is the set of vertices and E is the set of edges. We want to find a feedback vertex set, i.e., a subset $V' \subset V$ such that V' contains at least one vertex from every directed cycle in G, while minimizing the cardinality of the feedback vertex set $|V'|$. This problem is equivalent to the known graph theory algorithm, "Minimum Feedback Vertex Set", which is shown to be approximable within $O(log|V|loglog|V|)$ [19]. Breaking the cycles result in a directed acyclic graph (DAG). This graph is topologically sorted in Step 4.

In Steps 5 and 6, the gates are sized in topological oder. This ensures that, the victim-aggressor duality is taken into account. Since we are sizing in the order of noise dependence, the effects of sizing up a driver will be seen down-straem, on the nets that it has an effect on. As explained earlier, the noise graph consists of nets that are failing and that are close to failing. The topological sort approach makes sure that if any of the 'close to failing' nets start failing due to one of its up-stream neighbors being sized up, this is detected and addressed. At each vertex, a proper gate size from the cell library is chosen such that the optimality, area and timing constraints are satisfied. By optimality, we refer to the noise reduction vs. area trade-off discussed in subsection 2.1. This is taken into account by not increasing a gate size passed a predefined point of diminishing return. The objective function in the size-up process is the negative of noise slack on the net.

Algorithm: Size_up(v)
Input: Type V vertex v
Output: library cell to replace vertex v
begin
1 **while** optimality and $area\_slack(v_a)$ and $time\_slack(v_a)$ are within constraints
2 replace v with next larger same functionality cell in library
3 **if** $noise\_slack(v_a) \geq 0$
4 **return**
5 **return**
end

4 Results

In this section, we present results of our algorithm on three large designs. The circuits used for experiments are chip_1, which has 31489 nets, chip_2, which has 39200 nets and chip_3 with 165481 nets. All three designs are actual high performance ICs and the number of nets reflect the number of top level nets analyzed by the noise analysis tool. Coupled RC interconnects were extracted using a commercial extraction tool and the analysis was performed in the typical process corner. Each cell in the standard cell library used in these designs was pre-characterized for holding and switching driver models and receiver noise failure criteria[1]. Initial noise analysis is performed on these three designs after they have been optimized for delay and slew constraints. During gate sizing for noise reduction, we use the timing slacks obtained from static timing analysis as timing constraints.

Table 1 shows the noise reduction results and Table 2 shows some statistical information on the runs. From Table 1, we can see that number of nets that fail the noise criterion goes down significanty, as much as by 96%. The last two columns in Table 1 show the maximum and average peak noise voltage reduction.

Circuit	# of nets	# of failing nets		noise reduction	
		Initial	After opt.	max	avg
chip_1	31489	42	23	30%	11%
chip_2	39200	52	2	48%	14%
chip_3	165481	414	56	87%	16%

Table 1. Noise reduction results

Chip	↓ agg	# vertices		# edges			CPU	
		init	BL	init	el	BL	load	opt
1	22	84	79	39	5	23	45 s	118 s
2	8	90	88	12	0	8	54 s	96 s
3	22	602	549	217	23	60	198 s	582 s

Table 2. Some statistics

Table 2 presents the following information in column order: Number of aggressor gates that have been sized down, number of vertices in the initial graph, number of remaining vertices after Break_cycles, number of edges in initial graph, number of edges eliminated by sensitivity pruning, number of edges remaining after Break_cycles, CPU time to load the circuit and parasitic information, CPU time for gate sizing.

In Figures 4 and 5, we show the changes in noise peak voltages at receiver inputs and changes in noise slack values at receiver outputs. The values on the x and y axis are before and after gate sizing respectively, in both figures. The 45 degree line is the $x = y$ line.The region below the line represents improvement in noise in Figure 4 and degradation in noise in Figure 5. Figure 5 is additionally divided into four quadrants by the vertical and horizontal lines at $x = 0$ and $y = 0$. Each dot in these figures corresponds to a noise simulation, and each net has two noise simulations. One noise simulation is where the victim net is stable at ground and the aggressors are switching from low to high and the other is the reverse situation. It can be seen from Figure 4 that, noise on most simulations has been reduced and only on some simulations noise has increased slightly. The increased-noise nets are the sized-down aggressor nets and those few nets that were eliminated from the graph during the Break_cycles procedure. For the case of chip_3, 149 simulations had more noise after gate sizing than before gate sizing. However when we look at Figure 5, we can see only 20 simulations whose noise slack values went to negative from positive (quadrant 4). Further investigation of these nets show that these nets also failed noise analysis before gate sizing. Hence, if a net did not fail before gate sizing, it remained that way after gate sizing. This shows that our algorithm is successful in not introducing new noise problems while trying to fix the existing ones. This is due to the fact that our algorithm checks the noise on aggressors as they are sized down and also the aggressor-victim duality is taken into account properly through the topological sort approach. These figures also show that, noise failure on many nets has been improved although the net was not fixed. But on the other hand, some nets that were failing initially, end up failing worse after gate sizing. All such nets are the ones that have been eliminated during Break_cycles procedure. We minimize such nets by choosing the minimum number of vertices to be eliminated from the noise graph.

As a result, our algorithm reduced number of failing nets significantly $(45\%, 96\%$ and 86% in three designs, respectively) while not introducing any new failures. Some controlled noise increase on aggressor nets was allowed, making sure that they stayed within acceptable positive slack. Even with the increase of noise on eliminated vertex nets, average noise reduction was $(11\%, 14\%$ and $16\%)$ respectively for the three designs.

Finally, Figure 6 shows the percentage change in peak noise values for chip_1 and chip_3.

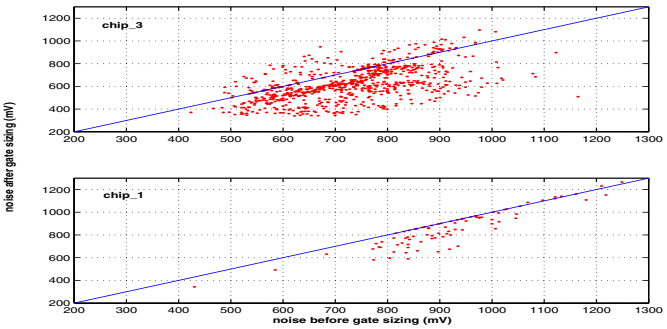

Figure 4. Noise peak changes at receiver input

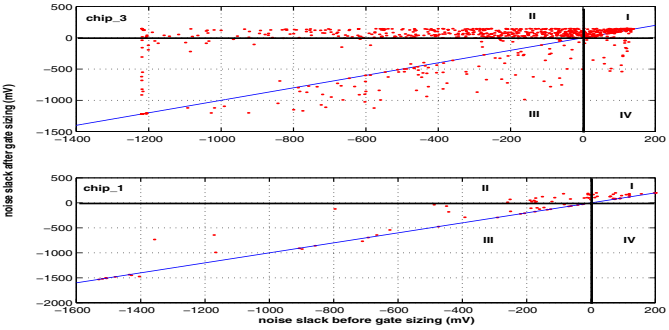

Figure 5. Noise slack changes at receiver output

5 Conclusion

In this paper, we presented a post-route gate sizing algorithm for crosstalk noise reduction. The algorithm is timing and area constrained and takes into account the victim-aggressor duality through a topologically sorted noise graph. The method utilizes sizing in both directions and has been shown to be effective on large high performance designs.

References

[1] S. Alwar, D. Blaauw, A. Dasgupta, A. Grinshpon, R. Levy, C. Oh, B. Orshav, S. Sirichotiyakul, and V. Zolotov. Clarinet: A noise analysis tool for deep submicron design. In *Proceedings of Design Automation Conference DAC*, pages 233–238, June 2000.

[2] K. L. Shepard and V. Narayanan. Noise in deep submicron digital design. In *Proceedings of ICCAD-96 Intl. Conference on Computer Aided Design*, pages 524–531, November 1996.

[3] S. Sirichotiyakul, D. Blaauw, C. Oh, R. Levy, V. Zolotov, and J. Zuo. Driver modeling and alignment for worst-case delay noise. In *Proceedings of Design Automation Conference DAC*, pages 720–725, June 2001.

[4] P. D. Gross, R. Arunachalam, K. Rajagopal, and L. T. Pileggi. Determination of worst-case aggressor align-

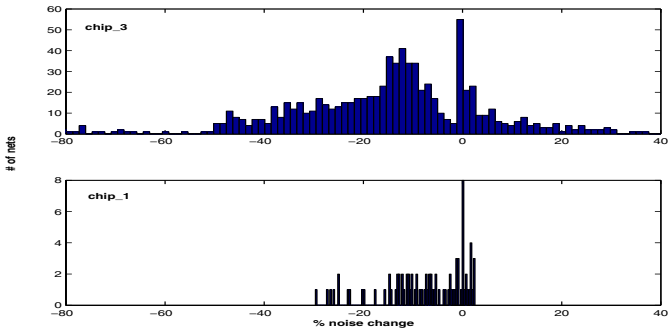

Figure 6. Percentage noise peak change

ment for delay calculation. In *Proceedings of the IEEE International Conference on Computer-Aided Design, ICCAD-98*, 1998.

[5] K. L. Shepard. Design methodologies for noise in digital integrated circuits. In *Proceedings of Design Automation Conference DAC*, pages 94–99, 1998.

[6] A. Vittal, L. H. Chen, M. Marek-Sadowska, K. P. Wang, and S. Yang. Crosstalk in VLSI interconnections. *IEEE Transactions on Computer Aided Design*, 18:1817–1824, December 1999.

[7] A. Vittal, L. Hui Chen, M. Marek-Sadowska, K. P. Wang, and X. Yang. Modeling crosstalk in resistive VLSI interconnections. In *Proceedings of International Conference on VLSI Design*, pages 470–475, January 1999.

[8] J. Cong, D. Zhingang, and P. V. Srinivas. Improved crosstalk modeling for noise constrained interconnect optimization. In *Proceedings of ASP/DAC Asia South Pasific Design Automation Conference*, pages 373–378, 2001.

[9] Murat R. Becer, David Blaauw, V. Zolotov, R. Panda, and Ibrahim N. Hajj. Analysis of noise avoidance techniques in dsm interconnects, using a complete crosstalk noise model. In *Proceedings of Design Automation Conference in Europe DATE*, pages 456–463, March 2002.

[10] H. Zhou and D. F. Wong. Global routing with crosstalk constraints. In *Proceedings of Design Automation Conference DAC*, pages 374–377, 1998.

[11] P. Saxena and C. L. Liu. Crosstalk minimization using wire perturbations. In *Proceedings of Design Automation Conference DAC*, pages 100–103, 1999.

[12] C. P. Chen and N. Menezes. Noise aware repeater insertion and wire sizing for on-chip interconnect using using hierarchical moment matching. In *Proceedings of Design Automation Conference DAC*, pages 502–506, 1999.

[13] C. J. Alpert, A. Devgan, and S. T. Quay. Buffer insertion for noise and delay optimization. In *Proceedings of Design Automation Conference DAC*, pages 362–367, 1998.

[14] I. H. R. Jiang, Y. W. Chang, and J. Y. Jou. Crosstalk driven interconnect optimization by simultaneous gate and wire sizing. *IEEE Transactions on Computer Aided Design*, 19:999–1010, September 2000.

[15] T. Xiao and M. Marek-Sadowska. Gate sizing to eliminate crosstalk induced timing violation. In *Proceedings of ICCD*, pages 186–191, 2001.

[16] T. Xiao and M. Marek-Sadowska. Crosstalk reduction by transistor sizing. In *Proceedings of ASP/DAC Asia South Pasific Design Automation Conference*, pages 137–140, 1999.

[17] M. Hashimoto, M. Takahashi, and H. Onodera. Crosstalk noise optimization by post-layout transistor sizing. In *Proceedings of ISPD*, pages 126–130, 2002.

[18] V. Zolotov, D. Blaauw, S. Sirichotiyakul, M. Becer, C. Oh, R. Panda, A. Grinshpon, and R. Levy. Noise propagation and failure criteria for vlsi designs. In *Proceedings of ICCAD-02 Intl. Conference on Computer Aided Design*, pages 587–594, November 2002.

[19] G. Even, J. Naor, B. Schieber, and M. Sudan. Approximating minimum feedback sets and multi-cuts in directed graphs. In *Proc. 4th Int. Conf. on Integer Prog. and Combinatorial Optimization*, pages 14–28, 1995.