Simultaneous Shield and Buffer Insertion
for Crosstalk Noise Reduction in Global Routing
∗

Tianpei Zhang and Sachin S. Sapatnekar
Department of Electrical and Computer Engineering, University of Minnesota
E-mail: zhangt, sachin@ece.umn.edu

Abstract
We present a method for incorporating crosstalk reduction criteria into global routing under an innovative power supply architecture, while considering the constraints imposed by limited routing and buffering resources. An iterative procedure is employed to route the signal wires, assign supply shields, and insert buffers so that both buffer/routing capacity and signal integrity goals are met. In each iteration, shield assignment and buffer insertion are considered simultaneously via a dynamic programming-like approach. Our noise calculations are based on Devgan’s noise metric, and our work shows, for the first time, that this metric shows good fidelity on average. Experimental results on testcases with up to about 10,000 nets point towards an asymptotic run time that increases linearly with the number of nets. Our algorithm achieves noise reduction improvements of up to 53% and 28%, respectively, compared to methods considering only buffer insertion, or only shield insertion after buffer planning.

1 Introduction
Interconnect performance issues have become dominant in determining the performance of a circuit. In addition to considering traditional metrics, it is important to integrate the analysis and optimization of interconnect crosstalk noise into routing in order to maintain signal integrity. Functional noise is seen when a victim net changes its level due to the switching of its neighbor aggressor nets, and this could lead to circuit malfunction. Delay noise is caused when the victim and aggressor nets switch at the same time, which causes the effective coupling capacitance to become unpredictable, thus affecting the delay.

Various noise estimation and avoidance techniques have been proposed over the years [2, 4, 5, 7–9, 12, 14]. However, there are several considerations that are not fully addressed in previous work. First, although power supply wires are used as shields in [4] and [8], with the increasing number of crosstalk-affected nets, routing congestion and routability are major concerns since supply wires will also compete for the limited available routing resources. Therefore, a realistic crosstalk-conscious router must consider the trade-off between routing resource consumption and noise reduction. Second, modern designs employ a large number of buffers to achieve timing closure [17]. As a side-benefit, buffers can also effectively reduce noise by recovering the noise margin [2]. However, next generation design will see a larger number of nets requiring more buffers, and it is projected that at 32nm technology, a very large proportion of all cells will be buffers [13]. Under limited silicon area, this will produce high contention for the limited buffer resources. Hence a buffer-only noise reduction may not meet the noise requirements due to the contention. This motivates our simultaneous buffer and shield insertion scheme for functional noise reduction. With the help of shields, buffers can effectively block noise propagation.

Our work considers the problem of crosstalk noise reduction during global routing under restrictions on the availability of routing and buffer resources. We simultaneously allocate power supply wires as shields and insert buffers in the global routing phase in order to route nets under a noise budget. To utilize existing power supply wires, this method is presented under the backdrop of a power/ground (P/G) network architecture. The procedure results in a signal/power co-routing solution at the global level. While it will primarily deal with the functional noise as defined above, the insertion of supply shields with stable voltage levels between signal wires also provides the subsidiary benefit of greatly easing delay uncertainties and therefore relieves the delay noise. We have also incorporated considerations to insert a sufficient number of buffers to control the delays and slew on each signal line.

Our method works iteratively: starting with an initial global routing solution, an enumerative dynamic programming-like algorithm is used to simultaneously assign supply shields and buffers to meet the noise budget for each net, one at a time, to find a minimum cost solution for the net. Next, an iterative rip-up-and-reroute step is performed to better meet the routing and noise goal. We simultaneously take into account the limitations on routing/buffer resources and the needs for signal integrity and provide a global routing solution that is immune to capacitive coupling noise. For comparison purposes, we also implemented an intelligent greedy approach which is faster, but less effective in resource allocation.

2 Preliminaries
2.1 Global Routing and Buffer Model
As shown in Figure 1, our global routing model tessellates the entire chip into an array of grid cells, referred to as the routing grid. Net \( N \) consists of a set of electrically equivalent pins \( \{ s, p_1, p_2, \ldots, p_k \} \) distributed in different routing grid cells, that must be connected by wires, of which \( s \) is the source and \( p_1, p_2, \ldots, p_k \) are the sinks. The dual graph of the routing grid tessellation is the routing graph \( G \), which is shown in Figure 1(a) as the dashed lines. Connections among all of the pins will be routed over the routing graph \( G \). Each edge in the routing graph corresponds to a boundary \( e_j \) in the routing grid that connects grid cells \( i \) and \( j \). The grid length \( L_c \) is defined as the center-to-center distance between two neighboring grid cells. Due to geometrical limitations on the boundary, we require that \( W_c \leq C_s \), in which \( W_c \) is the total width (including wire spacing) used by signal and power lines passing the boundary, and \( C_s \) is the geometrical width of the boundary \( c \), or the boundary capacity. Violation of this requirement results in boundary overflow. We follow a two-layer routing model in which horizontal and vertical lines are routed on different layers. The aim of routing is to eliminate boundary overflows while achieving other performance-related goals.

Our procedure also incorporates buffer insertion as an effective way to reduce delay and noise. We adopt the distributed buffer model proposed in [1], in which buffers are interspersed within the routing grids, and their exact location is undetermined until later in the design process. Figure 1(b) shows an inset view of a part of the routing grid where distributed buffers are inserted into a signal wire. For a grid cell \( i \), the number of buffers available is denoted as \( B_i \). If the number of utilized buffers is \( b_i \), then \( b_i \leq B_i \) must be satisfied, otherwise we have buffer overflow. To control the interconnect delay and slew rate, as in [1, 15], we enforce a constraint such that the maximum total in-

∗This work was supported in part by the NSF under award CCR-0098117.
terconnect length that can be driven by a buffer (gate) is the length of $M$ grids.

2.2 Power Supply Architecture

A traditional power supply architecture is composed of a regular dense grid that traverses the entire layout area. However, different parts of the layout require different amounts of current, and the density of the grid does not have to be uniform. This may be exploited for routing flexibility [10].

We assume that the power grid is an array of variable density, and the integrity of the supply grid is maintained by ensuring that the average and minimum number of wires feeding every block exceeds a threshold. The layout is divided into blocks, and for each block $i$, we are given:

- a Minimum Average Number (MAN) of supply wires $MAN_i$ per grid edge,
- a Minimum Number (MN) of supply wires $MN_i$ running over each grid edge belonging to the block $i$.

Note that $MAN_i$ and $MN_i$ are both defined per edge and together define a basic structure of power network, thus enabling the power grid to be incorporated even before a detailed power architecture is determined. Any extra power lines/shields beyond $MN_i$, and $MAN_i$ will only improve the power grid performance [10]. This model works well on intermediate metal layers like that of [11], where the variable number of power lines in adjacent blocks do not have to match exactly since they can be connected to each other through upper layers. All of the edge capacities are shared by signal wires and power supply wires.

If signal wires utilize too much routing capacity at a boundary, then it is possible to make enough room for P/G wires. We refer to the difference between the required and actual number of P/G wires in a block as the P/G shortage. The P/G wires in the supply grid are used not only to carry power currents, but also work as shields between aggressor and victim signal wires to reduce noise; we will use the terms supply wire and shield interchangeably. Also for the consideration of design for manufacturing, filling the remaining routing capacity left by signal routing with supply wires can improve manufacturability and performance predictability which may be deterioated by the Chemical-Mechanical Planarization (CMP) step in manufacturing [21].

2.3 Fidelity of Devgan’s Noise Metric

Devgan’s noise metric [5] is employed in this paper to find the capacitive coupling noise and the corresponding noise margin. This metric provides an upper bound for the crosstalk noise in an RC circuit, and its calculation is very similar to that of Elmore’s delay. An example of applying Devgan’s noise metric to noise estimation is shown in Section 2.4.2. However, this metric is known to potentially result in large overestimates [3]. In the context of routing multiple nets, we argue that it is the average error of a noise metric over many nets that is important rather than the maximum. If the average error of the noise predictor is relatively low, then the overall utilization of shielding and buffering resources will be good; for a pessimistic noise metric, a large average error will result in the over-utilization of resources.

2.4 Shield Insertion and Noise Calculation

2.4.1 Arrangement of shields

The insertion of a supply wire between two signal wires will shield them from each other, so that there will be no significant capacitive coupling noise between the lines. Moreover, the insertion of the constant-voltage supply wire will reduce the delay uncertainty of adjacent signal nets, as compared to the case when that signal wire is next to a simultaneously and oppositely switching signal wire. We say that a side of a signal wire is provided protection if it neighbors a supply shield. Figure 2 shows five aggressor nets and a two-sink victim net with two supply wires as shields.

In the global routing phase, the exact positions of signal nets are still undetermined and hence neighborhood information is not fully available. We attempt to determine a worst-case scenario based on the information that is available. We assume that over a routing edge, if one side of a signal net is not placed next to a shield, it will (pessimistically) be adjacent to an aggressor net that will induce coupling noise on the net. Thus a signal wire must have supply lines placed on both sides to be fully protected. However, due to limited routing resources, a net may not be fully protected. We refer to the number of protected sides of net $k$ at a specific routing edge $e_{ij}$ as $P_{ij, k}$, and this can take a value of 0, 1, or 2.

If $P_{ij, k}$ is known for the signal wires across edge $e_{ij}$, the number of supply wires required for shielding can be found as follows. If there are $S$ signal wires requiring protection on a single side, and $D$ signal wires requiring protection on both sides, we must have $p$ power supply as shields to achieve the protection, and
Noise calculation

A noise margin, \(NM_{\text{spec}}\), is specified for each gate or buffer input in the circuit, and represents the largest noise voltage that will not result in a circuit malfunction. The choice of \(NM_{\text{spec}}\) is based on the fidelity of Devgan’s metric, and can be selected by inflating the actually desired noise margin, and can be selected by inflating the actually desired

\[NM(i) = \min_{\text{all child node } j} (NM(j) - V_n(i \leftrightarrow j)),\]

where \(V_n(i \leftrightarrow j)\) is the noise voltage induced between \(i\) and \(j\). A net is noise free if at both the source and any buffer output,

\[I_n \cdot R_d \leq NM\]

Here \(R_d\) is the gate driver resistance, and \(I_n\) is the induced noise current, calculated using Devgan’s metric [5]. The noise reduction is illustrated by an example shown in Figure 3. A signal wire segment extends from the center of grid cell \(i\) to the center of its neighbor cell \(j\) with unit length coupling capacitance and resistance to be \(C_e\) and \(R_e\) respectively, and the aggressor voltage change rate is \(\mu\). Since the left side of the wire segment is shielded, only the right side aggressor will induce noise, giving us the following result according to Devgan’s noise margin:

\[NM(j) = NM(i) - ReLc(1/2CeLc\mu + In(i))\]

If the noise at \(j\) satisfies the constraint (4), the buffer will block the propagation of noise, and the noise margin at \(j\) will be recovered to \(NM_{\text{spec}}\), with noise current \(I_n(j')\) back to 0 as well.

Problem Formulation

The formal statement of the problem is as follows. Given a tiling of a chip and the corresponding routing graph \(G = (V, E)\), nets \(N = \{n_1, n_2, \ldots, n_m\}\), the edge capacity \(C_{ij}\) for every edge \(e_{ij} \in E\), and the buffer capacity \(B_i\) for each routing grid cell \(i \in V\), the problem is to find a routing solution that:

1. determines the routes for each net on the routing graph,
2. satisfies the power/ground density constraints, i.e., the average and minimum density \(MAN_i\) and \(MN_i\) of each block must be met,
3. determines the grid cells in which a net is to be buffered, subject to the buffer capacity constraint,
4. finds \(P_{ij, k}\) for each edge \(e_{ij}\) in the routing of net \(k\), subject to the edge capacity constraint (2),
5. satisfies noise constraint (4) for all nets.
6. ensures that the total amount of interconnects that can be driven by a buffer (gate) is at most \(M\) grid units.
structure of the net in a bottom-up manner, starting from the sinks and moving towards the source, moving along one grid square at a time. Each tree is described in terms of nodes that correspond to the grid cells that it passes through. Assigning a direction to the tree from the source to the sinks, we refer to the grid cell that contains the immediate predecessor (successor) of a given node \( n \) in the tree as the parent (child) cell of the grid cell containing \( n \). If a grid cell has more than two children, we can insert pseudo-nodes, so that the final tree is a binary tree for ease of later processing.

While traversing a net across a grid cell \( i \), we have two methods for protecting it from crosstalk noise:

1. By deciding whether to insert a buffer or not at grid cell \( i \): this corresponds to two possible buffer insertion configurations (0 or 1 buffer)
2. By protecting the net using supply shields on one side, on both sides, or choosing not to shield the net at all. If grid cell \( i \) is not the root of the tree, shield(s) may be inserted alongside the edge \( e_{ij} \) connecting current grid cell \( i \) and its parent grid cell \( j \) in the tree. This results in three possible configurations (0, 1, or 2 sides protected).

Therefore, in each bottom up step, we may have six possible configurations for the protection structure. However, we cannot locally determine at each grid point which scheme is globally optimal, and therefore an enumerative dynamic programming-like approach is adopted here in the same spirit as in Van Ginneken’s algorithm [18].

### 3.2.1 Protection Cost and Solution Architecture

While traversing a net bottom-up, at grid cell \( i \), we must find the protection cost corresponding to a protection structure, so as to measure the resource usage. For the protection cost related to buffer insertion, since the nets are processed one at a time, at any point in our insertion algorithm, the probability that an unprocessed net \( n \) crossing grid cell \( i \) will insert a buffer from \( i = 1/M \). Let \( p_i \) be the sum of these probabilities over all unprocessed nets crossing cell \( i \), and the cost for insertion of \( b_{eq} (= 0 \text{ or } 1) \) buffer at a specific grid tile \( i \) is similar to that in [1]:

\[
\text{cost buffer}_i (b_{eq}) = \begin{cases} 
0 & \text{if } b_{eq} = 0 \\
 b_{eq} & \text{if } b_{eq} = 1 \\
 \infty & \text{otherwise}
\end{cases}
\]

This cost function will significantly increase the cost penalty as buffer resources become contentious. For shielding cost, we can calculate the number of power supply wires required based on the number of sides to be protected \( N_{sh} \) and equation (1). In the same spirit to punish contentious resource usage, the shielding cost \( \text{cost}_{\text{shield}_i} \) for a signal wire can be obtained from a similar form of equation as (9) above, but the predicted shield usage takes a different approach: each unprocessed signal net will probabilistically have 1 side to be protected (assuming equal probability for \( N_{sh} = 0, 1, \text{or } 2 \)). Both the shielding cost and buffer cost are a measurement of the number of resources used, and they are approximately of the same order of magnitude. Therefore, we can combine them with a weighting factor \( \lambda \) (determined by the availability of resources) to develop a metric for resource usage, which we call the protection cost at grid cell \( i \), denoted as \( PC_i \): 

\[
PC_i (b_{eq}, N_{sh}) = \lambda \text{cost buffer}_i (b_{eq}) + \text{cost}_{\text{shield}_i} (N_{sh})
\]

where \( j \) is the parent grid cell of grid cell \( i \). This comprehensive cost function can be used as a metric to compare resource usages from different insertion schemes, and our goal is to find a minimum cost scheme satisfying the noise requirement, so as to resolve the contention for protection resources among nets.

At a cell \( i \) during the bottom-up traversal, the noise margin and noise current will vary according to the protection structure we choose, i.e., whether a buffer is inserted and the number of sides of the net that are shielded. If a buffer is inserted, the noise current will be “reset” to 0 and the noise margin set back to \( NM_{spec} \). At each unbuffered location, depending on the number of sides of a signal wire that are shielded, we can have the following from equations (5) and (6):

\[
\Delta I_n (N_{sh}) = (2 - N_{sh}) \cdot I_{Ce} \cdot C_{e} \cdot \mu
\]

\[
\Delta NM (N_{sh}) = R_{e} \cdot \left( \frac{1}{2} (2 - N_{sh}) L_{c} C_{e} \mu + I_{0} (i) \right)
\]

where \( \Delta I_n \) is the increase in the noise current due to the number of sides getting shielded \( N_{sh} \in [0, 1, 2] \), and \( \Delta NM \) is the noise margin decrease during the bottom-up step. To keep a record of the protection structure in the enumeration, we define a protection solution at routing cell \( i \) to be a 4-tuple \( S = \{ PC, NM, I_n, stru \} \), where \( NM \) is the noise margin at the end of the edge connecting grid cell \( i \) to its parent grid cell \( j \), \( I_n \) is the noise current induced by the neighboring signal wire at the same point, and \( PC \) is the protection cost of the current solution. The last component, \( stru = \{ buffer \}, N_{sh} \}, \) represents the protection structure of the solution, where \( buffer \) is a binary number representing the number of buffers utilized at grid cell \( i \), and \( N_{sh} \) is the number of sides (0, 1 or 2) on which the wire is protected.

If a solution \( S_1 \) is provably inferior to another solution \( S_2 \) at the same grid cell, i.e., if \( PC_1 > PC_2 \) or \( N_{sh1} > N_{sh2} \), then it is pruned from the set of solutions. To satisfy the constraint that the total amount of interconnect can be driven by a buffer (gate) at most \( M \) grid cells, we maintain a solution set array (SSA) of length \( M \) at each grid cell. Each element in SSA is a set of solutions, and the array is indexed from 0 to \( M - 1 \). The solutions in SSA[k], \( k \neq 0 \) correspond to a total downstream interconnect of \( k \) grid cells to the nearest downstream buffer(s), and SSA[0] stores the solutions that correspond to the insertion of a buffer at the current grid cell.

### 3.2.2 Protection Solution Building Algorithm

A unitary step in the enumeration algorithm is to build the solution set array at the current grid cell based on the arrays at its child cell(s). The pseudo code for the algorithm is listed in Figure 4. The main procedure in the algorithm calls function \text{Find sol set array} at the source cell, and returns the minimum cost solution that satisfies the noise constraint. In function \text{Find sol set array}, solutions in set SSA[k], \( k \neq 0 \) are propagated from solutions in the lower indexed sets of the children grid cells. The propagation is performed differently for one child and two children situations as shown in Figure 5 (a) and (b). During the propagation process, we update the noise margin and shielding information to form new solutions. At the same time, the least cost children solutions are selected and combined to build solutions in set SSA[0] in which buffer is inserted right at current grid cell. In the above steps, functions \text{Propagate}, \text{Insert buf} and \text{Merge} are called to build solutions. The solution pruning in the last steps will greatly reduce the solution set size. There are three types of pruning techniques employed here: the first discards those solutions that violate the noise constraints (4), the second discards the solutions that violate the buffer or wiring capacity, and the third removes any solution that is inferior to another solution in the same SSA.

While there is no concrete way of proving that the size of SSA will be small, the pruning technique works efficiently in practice: our experiments show that the number of solutions at each grid cell is limited between 3 and 60, and is less than 15 in most cases. We also observe that the asymptotic total runtime increases linearly with the number of nets in the benchmark circuits.

### 3.3 Step 3: Refinement

After the simultaneous shield and buffer insertion for noise reduction, refinement steps are applied if there are still some nets cannot be protected from noise constraint violation. The procedure is similar to that used in global routing phase. We rip-up and reroute all of the nets in the same fixed order as before. After one net is ripped up, it is rerouted immediately by the rerouting algorithm described in Section 3.1. However, the cost function in rerouting is now the combination of both the wiring congestion cost and the buffer congestion cost.

This will drive the net to go through regions where wiring capacity and buffer congestion cost are improved, and the noise reduction is greatly increased.
the nets have been ripped up, rerouted, and then protected, the whole refinement step will be performed again if there is still some noise violation. However, our experimental results show that there will not be much improvement after the third iteration. To provide additional protection from noise, in the last step, the unprotected nets will greedily take all of the unused wiring and buffer capacities along its path; however, this step is optional.

4 Experimental Results and Conclusion

Our algorithm is implemented in C++ on a Linux PC with a 2.8GHz CPU and 1GB memory. Out of the 12 benchmarks, the first ten benchmarks in Table 1 are obtained from the authors of [10]. The largest benchmarks synf and syn2 with over 10,000 nets are randomly generated.

We superimpose a grid over the the floorplan so that the geometry of each grid cell is almost a square. The number of buffers in each grid cell is generated randomly and the total number of buffers is listed in Table 1. We divide the design into several blocks, which correspond to different styles of circuits, such as control logic, data path, etc, and in our experiments, we use 7 blocks. The power supply requirements

---

1We did not use the ISPD98 placement benchmarks, because most of the nets in it are very short, which makes buffer insertion unnecessary, and cannot be used to illustrate the buffer contention problems that are projected for future technology nodes. This is consistent with the experience of the authors of [19].

2The noise threshold chosen here can be user-specified and is used to identify the nets with the largest noise, rather than as an exact predictor for the noise value. Since Devgan’s metric has fidelity but not accuracy, the nets violating this threshold will indeed be those with the highest noise.

Figure 4: Algorithm for building protection solution.

**Figure 5:** Updating SSA for one and two child grid cells. 

$M_{AN}$ and $MN$ are randomly generated but in a balanced manner across the chip (in practice, these will be dictated by the functional blocks). We also assume that a power grid wire is twice the width of a signal wire. The routing edge capacities are assigned as shown in the table, in the units of signal wire width. We assume that the grid length $L_g = 600\mu m$, that for all gates, the noise margin specification $\Delta NM_{pec} = 0.4V$ under a $V_{DD}$ of 1.8V, and that the aggressor voltage change rate $\mu = 9 \times 10^{11}$V/$\mu m$. The technology parameters used in the experiments are derived from [20] and [22] for the 0.18 $\mu m$ technology: unit length coupling capacitance $C_t=0.0583F/\mu m$, unit length resistance $R_t=373Î£/\mu m$, and buffer driver resistance $R_d=180Î£$.

We compare our results with those of two other methods. The first method is similar to the idea of [2], in which only buffers are inserted to reduce the noise, and the shielding effects are not considered. The buffers are also inserted by a dynamic programming-like algorithm, trying to achieve the noise constraint with the fewest number of buffers. The second method that we compared against is a greedy approach, in which buffers are first assigned in the same way as [1]. With the buffer positions known, we attempt to insert shield wires for each routing edge. For each net, the greedy shield insertion is composed of two steps:

1. We use a bottom-up approach, using every possible shield along the routing edges to meet the noise constraint.
2. If step 1 is successful, it may be the case that more than enough buffers have been inserted. Then we follow a top-down peel-off procedure to remove all of the unnecessary shields and buffers until the noise constraint or driving length constraint has been violated. The peel-off is greedy in the sense that a shield that is closer to the source of a tree will be removed greedily first. If there are multiple choices at any step in the top-down process, the branch with the higher noise margin will have its shield peeled off first.

Both of the above comparison methods employ the same routing and rerouting procedure as our approach.

The experimental results are listed in Table 1. The first eight columns show some basic properties of the circuits. Next, the results of our method which introduces buffers and shields (BS), the first method which introduces buffers only (B), and the greedy buffering and shielding method (G) are shown. Empirically, results show that the asymptotic run time of our buffer and shield insertion algorithm is linear in the number of nets, and our algorithm scales easily to cases with over 10,000 nets. The $B$ and $G$ columns show that these methods can result in noise protection failures on as many as 53% (circuit apte) and 28% (circuit armi33) of the total number of nets. In comparison, our simultaneous shield and buffer insertion approach has achieved the protection goals successfully without much sacrifice in speed, and in all cases, all of the nets meet the noise constraints.

The buffer-only approach shows poor performance because of the restricted number of buffers that are available. This will become more of an issue in future technologies, as projected by [13]. The greedy approach, on the other hand, performs buffer insertion and shield protection in separate steps, and no concerns of noise constraint are con-
sidered in buffer insertion, resulting in an inferior performance to our integrated buffer and shield insertion solution.

An additional advantage of our approach is the adaptive P/G architecture, which enables a flexibility between the requirements of signal and power routing, so that both routing and P/G requirements are simultaneously met. For all three algorithms in Table 1, the routing overflow are almost 0 for all benchmarks, and is hence not listed. However, in cases where more nets must be protected to resolve the remaining noise violations, extra routing resources must be employed, leading to overflows. Table 2 reports the overflow results for our algorithm and the greedy algorithm if 100% protection is desired (the buffer-only algorithm does not use shield resources, and is omitted). The results show that much more routing resources have to be sacrificed to obtain a good protection for the greedy algorithm, while our algorithm can successfully achieve a good protection without extra routing overflow.

Due to the pessimistic nature of Devgan’s metric, our buffer and shield insertion algorithm may over-estimate and use more than enough protection resources to accomplish full protection, or under stringent protection resources, may result in false-fails. To compensate for the pessimism of Devgan’s metric, we heuristically inflate the actual specified noise margin in practice. If chosen carefully, the inflated noise margin as input to our algorithm will generate protection solutions that require fewer protection resources, but still satisfy the original noise margin requirement. For example, we have inflated the specified noise margin from the actual 0.4V to be 0.5V and 0.6V respectively. The protection solutions are simulated with SPICE, and the results are listed in Table 3. Due to long run times, we randomly selected up to 700 nets from each circuit for simulation; for smaller benchmarks such as hp, all nets were simulated. As can be seen, with $N_{M_{spec}}$ inflated to be 0.5V, almost 100% of the solutions can still satisfy the original 0.4V noise margin; while this percentage drops to about 90% when $N_{M_{spec}}$ is inflated to 0.6V. Practically, we may choose to inflate $N_{M_{spec}}$ by about 25% to acquire a good yet economic solution.

<table>
<thead>
<tr>
<th>Circuit</th>
<th># of nets</th>
<th>Available buffers</th>
<th>EC</th>
<th>Grid</th>
<th>$M$</th>
<th>Average</th>
<th># noise violation nets</th>
<th>Run time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ami35</td>
<td>112</td>
<td>3011</td>
<td>9</td>
<td>30 × 33</td>
<td>6</td>
<td>5.2</td>
<td>1.2</td>
<td>0</td>
</tr>
<tr>
<td>ami49</td>
<td>368</td>
<td>6889</td>
<td>16</td>
<td>30 × 33</td>
<td>7</td>
<td>4.2</td>
<td>1.5</td>
<td>0</td>
</tr>
<tr>
<td>hot</td>
<td>77</td>
<td>1811</td>
<td>9</td>
<td>30 × 33</td>
<td>3</td>
<td>3.4</td>
<td>1.7</td>
<td>0</td>
</tr>
<tr>
<td>playout</td>
<td>1294</td>
<td>15884</td>
<td>56</td>
<td>30 × 33</td>
<td>7</td>
<td>18.0</td>
<td>5.5</td>
<td>0</td>
</tr>
<tr>
<td>a9c3</td>
<td>1148</td>
<td>11847</td>
<td>44</td>
<td>30 × 33</td>
<td>6</td>
<td>13.0</td>
<td>5.6</td>
<td>0</td>
</tr>
<tr>
<td>ac3</td>
<td>200</td>
<td>4034</td>
<td>14</td>
<td>30 × 33</td>
<td>6</td>
<td>6.4</td>
<td>2.4</td>
<td>0</td>
</tr>
<tr>
<td>hc7</td>
<td>430</td>
<td>7938</td>
<td>26</td>
<td>30 × 33</td>
<td>7</td>
<td>7.8</td>
<td>4.0</td>
<td>0</td>
</tr>
<tr>
<td>n200b</td>
<td>1714</td>
<td>16903</td>
<td>65</td>
<td>33 × 33</td>
<td>6</td>
<td>19.6</td>
<td>9.4</td>
<td>0</td>
</tr>
<tr>
<td>n300</td>
<td>1893</td>
<td>23295</td>
<td>68</td>
<td>33 × 33</td>
<td>7</td>
<td>19.9</td>
<td>8.4</td>
<td>0</td>
</tr>
<tr>
<td>syn1</td>
<td>10086</td>
<td>87773</td>
<td>334</td>
<td>40 × 40</td>
<td>6</td>
<td>96.5</td>
<td>60.7</td>
<td>0</td>
</tr>
<tr>
<td>syn2</td>
<td>10486</td>
<td>90577</td>
<td>348</td>
<td>40 × 40</td>
<td>6</td>
<td>102.8</td>
<td>61.3</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 1: Comparisons of routing and noise protection results. EC is the edge capacity; $BS$ represents the simultaneous buffer and shield insertion algorithm; $G$ represents the greedy algorithm; $B$ represents the buffer-only algorithm.

show that this method can route nets to meet both capacity and noise constraints. It is more effective than noise reduction using a buffer-only approach or a greedy approach.

References


<table>
<thead>
<tr>
<th>Circuit</th>
<th>$P_{0.5}$</th>
<th>$P_{0.6}$</th>
<th>Circuit</th>
<th>$P_{0.5}$</th>
<th>$P_{0.6}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ami35</td>
<td>100%</td>
<td>93.0%</td>
<td>ac3</td>
<td>97.4%</td>
<td>95%</td>
</tr>
<tr>
<td>ami49</td>
<td>100%</td>
<td>90.6%</td>
<td>hc7</td>
<td>100%</td>
<td>93.8%</td>
</tr>
<tr>
<td>hot</td>
<td>100%</td>
<td>81.3%</td>
<td>n200b</td>
<td>100%</td>
<td>95.0%</td>
</tr>
<tr>
<td>playout</td>
<td>97.6%</td>
<td>91.1%</td>
<td>n300</td>
<td>100%</td>
<td>92.3%</td>
</tr>
<tr>
<td>a9c3</td>
<td>99.7%</td>
<td>96.1%</td>
<td>syn1</td>
<td>100%</td>
<td>94.1%</td>
</tr>
<tr>
<td>ac3</td>
<td>100%</td>
<td>93.1%</td>
<td>syn2</td>
<td>100%</td>
<td>95.2%</td>
</tr>
</tbody>
</table>

Table 3: Protection rate with inflated $N_{M_{spec}}$. $P_{0.5}$ and $P_{0.6}$ are the percentage of nets getting fully protected under SPICE simulation with inflated $N_{M_{spec}}$ at 0.5V and 0.6V respectively.

We have shown in this paper a method for simultaneously inserting supply shields and buffers during global routing to reduce crosstalk noise under a novel power supply architecture. Experimental results