Electromigration Avoidance in Analog Circuits: Two Methodologies for Current-Driven Routing*

Jens Lienig
Robert Bosch GmbH
Reutlingen, Germany
jens@ieee.org

Göran Jerke
Robert Bosch GmbH
Reutlingen, Germany
goeran.jerke@de.bosch.com

Thorsten Adler†
sci-worx GmbH
Hanover, Germany
thorsten.adler@sci-worx.com

Abstract

Interconnect with an insufficient width may be subject to electromigration and eventually cause the failure of the circuit at any time during its lifetime. This problem has gotten worse over the last couple of years due to the ongoing reduction of circuit feature sizes. For this reason, it is becoming crucial to address the problems of current densities and electromigration during layout generation. Here we present two new methodologies capable of routing analog multi-terminal signal nets with current-driven wire widths. Our first approach computes a Steiner tree layout satisfying all specified current constraints before performing a DRC- and current-correct point-to-point detailed routing. The second methodology is based on a terminal tree which defines a detailed terminal-to-terminal routing sequence. We also discuss successful applications of both methodologies in commercial analog circuits.

1. Introduction

The recent booming market share for large-scale analog and mixed-signal circuits in automotive, telecommunication, consumer and computer applications has resulted in a significant increase in the complexity of these circuits. Despite this increase in complexity, these circuits are often painstakingly designed and laid out by hand. A primary reason for the lack of automation is the vast amount of expert knowledge typically required to meet constraints such as symmetry, current densities (including electromigration), voltage drops, temperature gradients, etc.

Despite these problems, there has been recently an increase in attempts to make custom analog layout tools a practical reality [14]. Here we present two new routing methodologies which address for the first time the problems of current densities and electromigration in analog circuits.

Unlike digital circuits, analog circuits must handle a multitude of different current levels, including extremely large currents in some applications (such as automotive circuits). Hence, the interconnect must be designed with the current that will be imposed on it in mind. Interconnect with an insufficient width (we assume the height to be a constant as given with many processes) may be subject to electromigration and eventually cause the failure of the circuit at any time during its lifetime [1],[4],[16]. Unfortunately, there does not exist any commercial routing tool which considers current densities during routing of analog signal nets.

A current-driven router must address the problem of unknown currents in net topologies during a sequential routing process. For example, the width of a net connecting only two terminals can be easily derived from the terminals' currents. However, if a third terminal is subsequently connected with this net using a Steiner point, the current would change and might make the previous route obsolete (e.g., if the wire cannot be widened). Generally speaking, any new connection to a previously routed sub-net may alter the currents imposed on the sub-net's paths and hence alter the correctness of its topological layout.

To the best of our knowledge, current-driven routing has been applied so far only to routing of power and ground nets in digital circuits, where the problem of unknown currents is addressed with a separate post-processing step that includes layout modifications [3],[7],[11],[12],[13],[15]. While this is feasible in power and ground routing due to its planar nature, limited number of nets and (still) unoccupied layers, current-driven routing of signal nets requires a different approach. Theoretical solutions might range from pure source-to-sink routing (and thus avoiding any links to previously routed segments of the same net) to a worst-case routing, where all connections are first routed with the maximum width and later reduced to the current-correct size.

We believe a practical solution lies somewhere between the extremes of these two approaches. In this paper we present two methodologies for current-driven routing that have been successfully tested in industrial design flows. A first approach separates the routing phase into two steps, Steiner tree routing and detailed routing. During Steiner tree routing, the estimated routing path of a net is determined by calculating the position of Steiner points. Since currents have already been considered

* Published in *Proceedings of 7th Asia and South Pacific Design Automation Conference (ASP-DAC) and 15th International Conference on VLSI Design*, January 2002, pp. 372-378.

† This work was done while T. Adler was with the Institute of Microelectronic Systems, University of Hanover.

during this global routing phase, the detailed routing can then be limited to two-point routing, with known currents at both end points, thus avoiding the above mentioned problems. Our second methodology is based on a terminal tree, which defines a detailed terminal-to-terminal routing sequence with known terminal currents.

Our paper is structured as follows: First we present the overall design flow, our current characterization method and the equations we used to calculate the wire widths. Then we describe both our Steiner tree global routing and our second methodology, terminal tree global routing. Next, we present details of our detailed routing method (applied for both global routing methodologies). We finish this paper with some experimental results.

2. Design Flow

The design flow of our approach is illustrated in Figure 1.

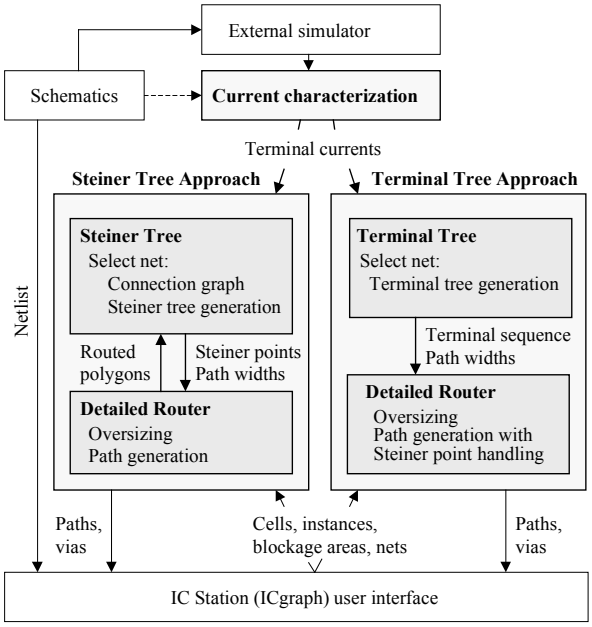

Figure 1. Design flow

During current characterization, current values attached to each terminal are obtained. These values are transferred to the routing tools either as part of the schematic netlist or as an ASCII file.

Our current-driven routing approaches have been integrated into the Mentor Graphics IC Station environment which reads the netlist from the schematic tool. After the initial placement of the cells is generated, the main layout components (cells, instances, blockage areas and nets) are forwarded to the routing tool.

The first methodology consists of Steiner tree routing followed by a detailed routing phase. During Steiner tree routing, a connection graph of the next net to be routed is generated. A Steiner tree is then established which represents both a valid topological route and a current-correct design. The resulting Steiner points and the calculated path widths are transferred to the detailed router. Here all layout elements are oversized in accordance with the wire width of the interconnection to be routed next. Afterwards, a point-to-point (Steiner point or terminal point) path generation is performed. The routing path, which has been initially generated with a "default" width, is then widened to the current-correct size. (No further layout modifications are needed due to prior oversizing.)

The second methodology essentially uses the same detailed router. However, instead of splitting up nets to "Steiner point level," it determines an optimized terminal-to-terminal routing sequence with known terminal currents. If a Steiner point is encountered during detailed routing, its "validity" is first checked by a re-calculation of the maximum currents occurring on the wires attached.

Both routing methodologies return the generated paths and vias to the main layout tool (IC Station).

3. Current Characterization

We utilize two approaches for the determination of realistic current values for each terminal. One method uses a standard circuit simulator for simulation of the circuit netlist ignoring parasitic wiring resistances. The second approach uses current values manually attached to the terminals in the schematic netlist by the designer.

The results from one or more simulations are post-processed by calculating a set of current vectors satisfying Kirchhoff's current laws. They represent a snapshot of the circuit's operation at the time of minimum and maximum currents at each terminal. This reduces the simulation results to a set of "worst case" current vectors. For a net with n terminals this may lead to up to $2n$ current vectors. Hence, up to $2n$ current values are attached to each terminal.

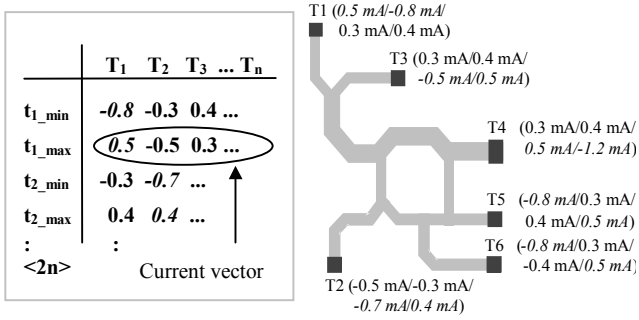

Figure 2. Current values assigned to terminals are their respective minimum/maximum values (shown in *italic*) and the current values at the other terminals' minimum and maximum point of time. Every current vector satisfies Kirchhoff's current law, i.e., its current sum is zero. (Only four current values are assigned per terminal for simplicity.)

Additionally, every terminal is labeled with its root mean square (RMS) current value which is derived from *all* simulation values. (The RMS current is responsible for *Joule heating* which greatly influences the effects of electromigration.)

4. Wire Width Determination

For a given temperature T_{ref}, the minimum wire width w_{min} is derived from the maximum and the root mean square current value as follows:

$$w_{min} = max \begin{cases} \dfrac{I_{eq} \cdot s}{d_{Layer} \cdot J_{\max}(T_{ref})} & (1) \\ \dfrac{|I_{max}| \cdot s}{J_{peak_Layer} \cdot d_{Layer}}, & (2) \\ w_{min_process} & (3) \end{cases}$$

where

I_{eq} = the root mean square (RMS) current on this path,
s = safety factor ($s \approx 1.1 .. 1.2$),
d_{Layer} = thickness of routing layer,
$J_{max}(T_{ref})$ = maximum current density allowed by this manufacturing process for temperature T_{ref}($J_{max}(150\,°C) \approx 1 .. 2$ mA/µm^2),
I_{max} = the maximum current on this path,
J_{peak_Layer} = layer dependent peak current density (process dependent),
$w_{min_process}$ = minimum wire width determined by manufacturing process.

The safety factor s is used to account for (1) terminal currents not caught during simulation, (2) small deviations of d_{Layer} due to process variations and (3) reduced accuracy due to our limitation to the maximum/minimum currents at each terminal.

5. Method 1: Steiner Tree Global Routing

5.1. Connection Graph

A connection graph G_c, similar to the one presented in [17], is used for layout representation during Steiner tree routing. G_c can be obtained by generating horizontal and vertical lines through all terminals of the net to be connected. Additionally, all horizontal and vertical edges of each obstacle encountered are extended until another obstacle or the boundary is reached (Figure 3). It has been shown in [10],[17] that the shortest path between two terminals in rectilinear metric is a path on the connection graph G_c.

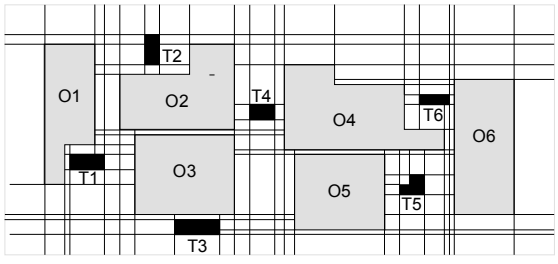

Figure 3. Construction of a connection graph

Using G_c we developed a line-search algorithm in order to find a path of minimum length between two points on this connection graph. Our approach is similar to the one presented in [17]. It is basically a line-search version of the Minimum Detour (MD) algorithm [6] with a generalized detour number concept.

5.2. Steiner Tree Generation

Generally, the current flow on wires connecting two Steiner points (e.g., ST1 and ST2 in Figure 4) is unknown prior to topology construction and has to be computed afterwards in order to widen all wires according to the currents imposed on them. However, this may lead to improper layouts due to design rule violations. In the example shown in Figure 4, obstacles O2/O3 and O4/O5 would have to be moved to generate the final layout.

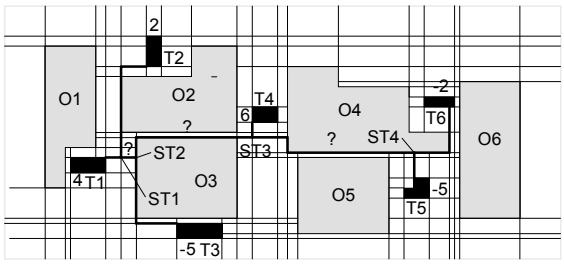

Figure 4. A conventional Steiner tree topology (Steiner points ST1 – ST4) with three current sources (T3, T5, T6) and three current sinks (T1, T2, T4, indicated with positive current values). Branches with unknown currents are marked with "?". Attached current values are only examples, in reality all current values must be considered.

In order to compute the unknown wire widths concurrently during Steiner tree construction, the Steiner tree must be built in a greedy, one-directional fashion.

Our algorithm calculates the Steiner tree layout by repeatedly computing an optimum Steiner point for three vertices at a time.

At first *calc_steiner()* computes the optimum Steiner point for the first three terminals using a modification of the strategy presented in [9]. After that Steiner point has been found, *detour()* is used to connect the first and the second terminal to the calculated Steiner point. Afterwards, *calc_steiner()* is called repeatedly to connect the last found Steiner point to the next two unconnected terminals, etc. The remaining two terminals are then connected to the last Steiner point calculated using *detour()*.

```
SteinerTreeRouting()
{
   label all terminals in increasing x-order
   i:=0; source:= terminals[i]; source_width:= width[i]
   for (;i<num_of_terminals-2; i++)
   {
      steiner_point:= calc_steiner(source, terminals[i+1],
                        terminals[i+2], source_width,width[i+1],
                        width[i+2])
      detour(source, steiner_point, source_width)
      detour(terminals[i+1], steiner_point, width[i+1])
      source:= steiner_point
      source_width:= source_width+width[i+1]
   }
   detour(source, terminals[i], width[i])
   detour(source, terminals[i+1], width[i+1])
}
```

The algorithm is a special variant of the rectilinear Steiner tree (RST) problem which is known to be NP-complete [5]. Our algorithm can be solved in $O(n \cdot m \cdot (l+m) \log l)$ time where n is the number of Steiner points to be connected, m is the number of candidate Steiner points evaluated during Steiner point search, and l is the number of obstacle edges.

Using this greedy Steiner tree construction, the Steiner tree planner is able to compute "on the fly" the unknown current flow in connections between two Steiner points by simply adding up the current values of the two wires connecting with the Steiner point.

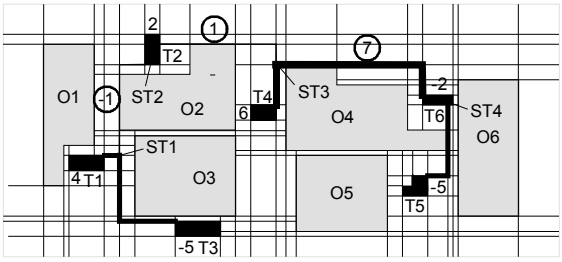

Figure 5. Our Steiner tree global routing solution avoiding post-routing modifications of the routing problem in Figure 4. The numbers in circles are current flows between Steiner points (unknown prior to Steiner tree routing) that have been taken into account already *during* Steiner tree generation.

6. Method 2: Terminal Tree Global Routing

A current-driven detailed router must solve the problem of altering current strengths in a prior routed sub-net whenever a new terminal is linked to it. In order to allow for a current calculation based on Kirchhoff's current laws prior to detailed routing, *at least the sequence of all terminals to be connected must be known*. Added detailed routing connections which directly link a "new" (not yet connected) terminal with its respective target terminal will then have no influence on current strengths calculated in the prior routed sub-net (with the calculation based on all terminals).

Hence, the most "coarse grain" approach possible for current-driven routing without post-routing layout modification is based on a pre-defined terminal-to-terminal routing sequence. Our second methodology investigates this approach, i.e., instead of using a Steiner tree to split up nets into "fine grain" two-point segments (as done in our first methodology) here only the terminal-to-terminal routing sequence is defined during global routing.

The routing sequence of terminals is based on a terminal tree which has been derived from the terminal locations and their respective distances relative to each other. The terminal tree is established in a one-directional fashion by iteratively connecting the nearest "unconnected" terminal with the set of already connected terminals. This terminal tree not only enables a near-optimum routing tree (based on distances) to be generated, but also provides a straightforward method of calculating the branch currents by applying Kirchhoff's current laws.

```
TerminalTreeRouting()
{
   label all terminals in increasing x-order
   i:=1; source:= terminal[i]
   for (i:=2; i≤num_of_terminals; i++)
   {
     for terminal[i]: find nearest terminal among terminals[1..i-1]
   }
   derive terminal tree
   from leaves inbound: calculate currents I of branches
   from leaves inbound: route terminal-to-terminal connection
}
```

A simple routing example of our terminal-tree-based approach is shown in Figure 6.

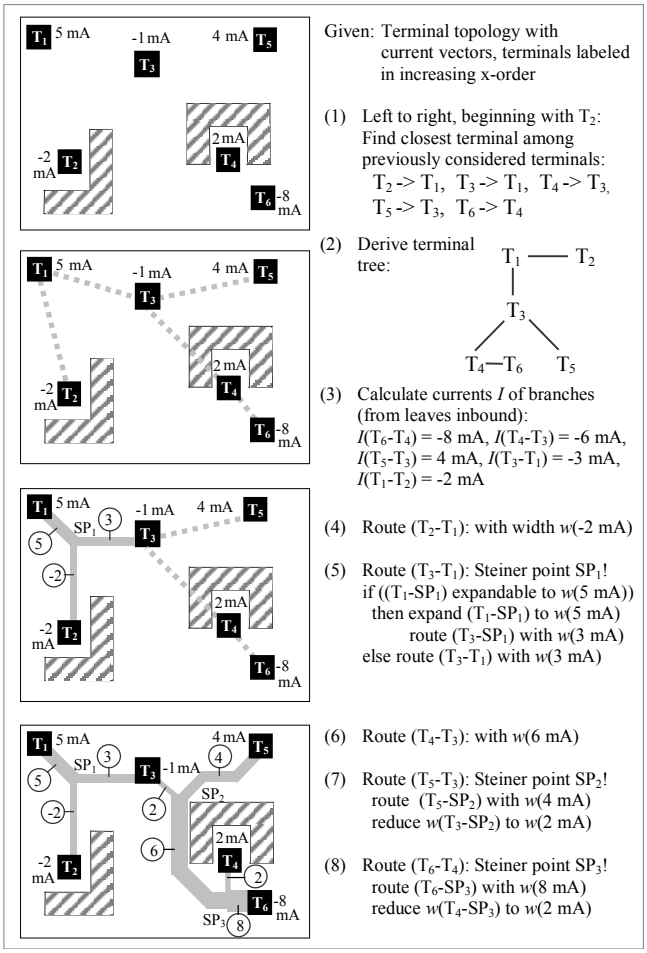

Figure 6. Graphical illustration of our second methodology based on a terminal tree. Steps 4 to 8 are presented only for illustration, they are part of the detailed routing process (see Section 7). Depicted current values are only examples, in reality all current values must be considered. Circled numbers symbolize the strength of current flows (in mA).

7. Detailed Routing

During detailed routing the exact paths and layer allocations of the wire segments are determined. These wire segments are simple two-point connections, linking either terminals and Steiner points (methodology 1) or only terminals (methodology 2). Furthermore, current-correct wire sizing has also been performed, i.e., any two-point connection "knows" its current-correct wire width.

The presented detailed routing strategy has been specifically developed in order to address numerous special characteristics of analog circuit routing as well as to support an efficient implementation of current-driven wire widths.

7.1. Oversizing

Our path search algorithm (see Section 7.4.) uses a fixed routing width in order to allow time-efficient path generation. Therefore, all polygons have to be oversized prior to each route. This strategy guarantees that the calculated "default-width path" can be replaced by a path of width *w* without violating any design rules.

The amount of oversizing *s* is given by

$$s = d_{min} + w/2 \qquad (4)$$

where d_{min} = minimum spacing on this layer, and *w* = current wire width.

7.2. Direction-dependent Spacing Rules

A current-driven router has to favor 45-degree routing because the current density at 45-degree corners is significantly reduced when compared to 90-degree corners. Using 45-degree routing requires distance rules that are direction dependent (with a larger spacing between diagonal wires). Furthermore, all corner coordinates have to be on a "manufacturing grid" with the consequence that diagonal wires must be widened to bring the corners of their edges on the grid.

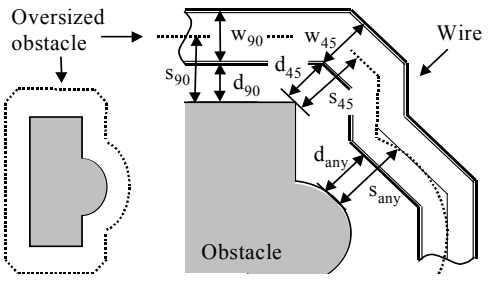

Figure 7. Transforming 90-degree angles to 45-degree angles during sizing (left) and the use of direction-dependent sizing values (right) [18]. Note that the 90- and 45-degree outline of the sized obstacle becomes the center line of the wire.

The use of diagonal path segments can be accommodated by using a sizing procedure which transforms 90-degree angles to 45-degree angles (Figure 7, left). The use of direction-dependent sizing values s_{90}, s_{45}, s_{any} enables direction-dependent distances and wire widths with $d_{90} < d_{45} < d_{any}$ and $w_{90} < w_{45}$ (Figure 7, right) [18].

7.3. Sizing of Arbitrary Polygons

Many tools for analog circuit design suffer from inadequate handling of polygons with arbitrary angles. We have developed a polygon-specific sizing strategy that guarantees the fulfillment of *direction-dependent* distance rules without wasting any space.

Polygons are sized by their corners, where each corner is replaced by one or more corners of an octagon (Figure 8). The number of octagon corners used is determined by the angles of the corresponding edges. The size of the octagon is derived from the direction-dependent sizing values s_{90} and s_{45}. (Standard sizing algorithms with one oversizing value have to use s_{45} or s_{any} for sizing in order to fulfill all distance rules. This leads to a waste of space in orthogonal directions since $s_{90} < s_{45}$.)

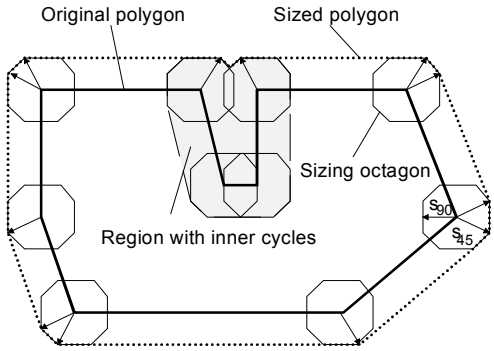

Figure 8. Oversizing of an arbitrary polygon using octagons to accommodate direction-dependent distance rules

7.4. Final Path Generation

We have developed a path search algorithm that combines a maze routing strategy (similar to [8]) with search area restrictions – so called "tunnel polygons." These tunnel polygons significantly reduce the search space (and run time) by restricting the search to an area derived from the appropriate wire geometry established during global routing.

It is known that the distance D between two points (x_1, y_1) and (x_2, y_2) is defined as

$$D = \sqrt[n]{|x_2 - x_1|^n + |y_2 - y_1|^n} \qquad (5)$$

with $n=1$ representing Manhattan metric and $n=2$ representing Euclidean metric. In order to realize HVD (horizontal, vertical, diagonal) routing, orthogonal neighbors in the search grid should have a distance of $D=1$ and diagonal neighbors $D=\sqrt{2}$. Hence, we label orthogonal neighbors of a grid point *i* with *i+2* and diagonal neighbors with *i+3*. This leads to a wave front nearly shaped as a circle.

After a point-to-point connection has been established, the current-correct wire width *w* is implemented by widening the routing path on each side by *w/2*. Layer changes are realized by vias extracted from a pre-sorted list according to the wire width.

7.5. Steiner Point Handling

Special consideration must be given to new connections linking to any other point than the target point, such as new Steiner points (as in Figure 6, connection (T_3-T_1)). Whenever a Steiner point (or any other point of the same net other than the target point) is encountered during path generation, the algorithm must re-calculate the maximum current through the previously routed net in order to determine if the Steiner point is feasible (i.e., if the previous route with the new Steiner point can be re-used for the new connection, as shown in Figure 9 (b)). If subsequent wire widening is needed but cannot be performed, then the respective net segment (on which the Steiner point would have been placed) is excluded from the current path search (Figure 9 (c)).

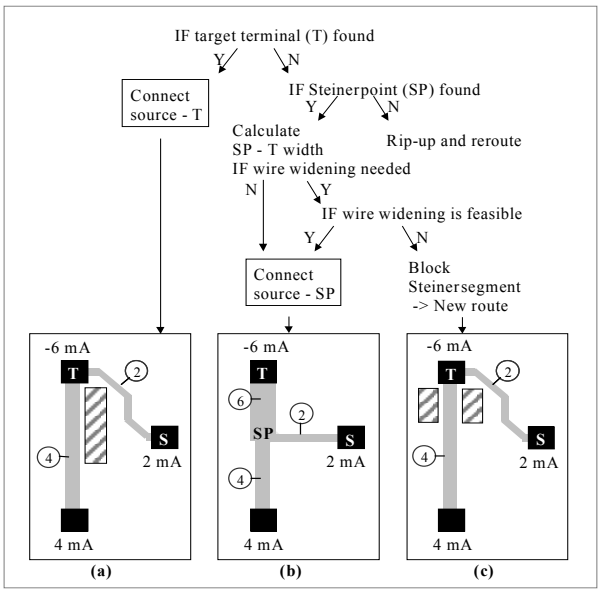

Figure 9. Path generation between source (S) and target (T) terminal with different strategies when encountering Steiner points (SP). Note that attached currents are only symbolic values, in reality all current values must be considered every time a new connection is established.

7.6. Connecting Arbitrary Polygons

Routing polygons of different widths combined with various cell layouts available in analog circuits may cause specific design rule violations in the region where the wire polygon connects to the cell polygon. Examples of two possible DRC violations are shown in Figure 10.

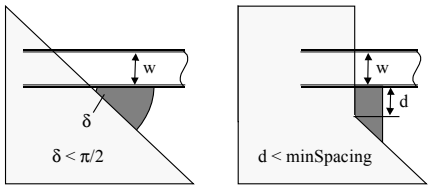

Figure 10. Possible design rule violations

These and similar design rule violations can be avoided by defining connectable regions on the source and target polygon as indicated in Figure 11. Arrows symbolize allowed directions for end segments of wires. The "width" of the connectable region depends on the current-correct wire width *w* of the net segment to be connected: Since wires are considered by their center lines, the width of a connectable region is reduced by *w/2* on both ends.

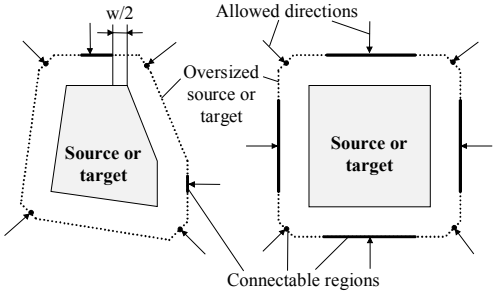

Figure 11. Defining connectable regions and directions around source and target polygons

8. Implementation and Results

Both methodologies have been integrated into the Mentor Graphics IC Station environment. The algorithms were implemented using C/C++ and the Mentor Graphics built-in programming language AMPLE.

Several commercial analog circuits (ranging from 90 to 380 devices per cell) were routed using both methodologies (Table 1). To ensure a proper comparison, we used only circuits that have been previously manually routed and current density adjusted by experienced designers.

Table 1. Characteristics of some commercial circuits routed with both methodologies

Circuits	Devices	Terminal-to-terminal connections ("flylines")	Nets
analog1	90	116	68
analog2	132	192	52
analog3	132	220	96
analog4	176	266	100
analog5	380	370	174

The results of both methodologies are compared in Table 2. It can be seen that both approaches deliver similar routing lengths and numbers of vias. (The relatively high number of Steiner points in the Steiner-tree based approach is due to the Steiner tree routing algorithm which favors the creation of Steiner points.) The routing area consumption of both approaches is slightly less when compared to layouts that have been manually routed and current-density adjusted, with the Steiner-tree-based approach providing a slightly larger reduction.

Table 2. Comparison of routing results

Circuit	Method	StP*	Routing lengths (μm)	Vias	Area Reduction**
analog1	Steiner tree	48	24342	142	-0.4 %
	Terminal tree	14	23921	138	0.0 %
analog2	Steiner tree	122	44710	138	-0.6 %
	Terminal tree	23	44714	144	-0.5 %
analog3	Steiner tree	116	37201	266	-0.3 %
	Terminal tree	27	36403	252	-0.1 %
analog4	Steiner tree	132	26302	188	-1.0 %
	Terminal tree	24	26748	186	-0.6 %
analog5	Steiner tree	234	45585	458	-1.1 %
	Terminal tree	44	45548	456	-0.9 %

* *StP = Number of generated Steiner points*
** *Area Reduction = Routing area compared to manual design*

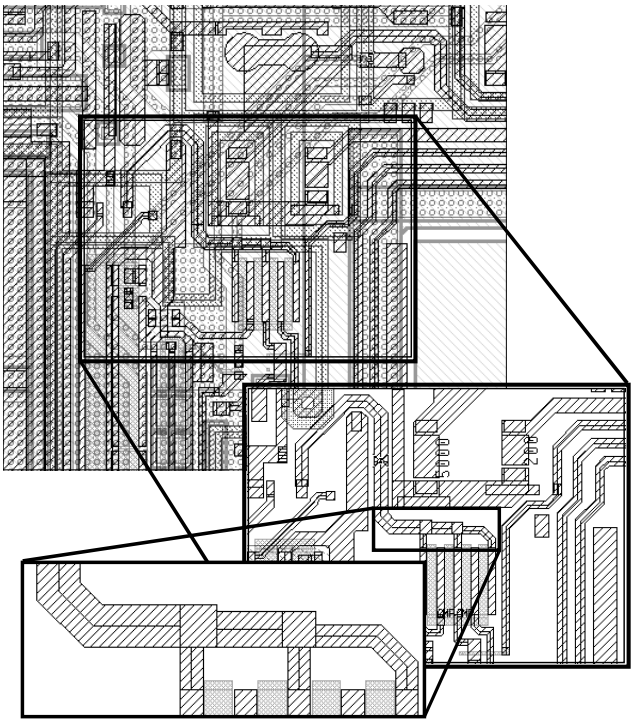

Figure 12. Excerpt of the routed circuit "analog5"

The presented approaches can be used in either semi-automatic mode (with specific nets selected for routing) or full-automatic mode (where all nets within a window are routed). Analog circuit designers usually prefer the first method, hence, run times vary widely. Using an UltraSPARC 10 workstation, routing of specific nets is performed within seconds; full-automatic routing of the entire layout requires a run time of minutes (up to one hour) with slightly shorter run times for the terminal-tree-based approach. (In comparison, experienced designers used one to two days to manually route and current-density adjust these circuits.)

In summary, while the terminal-tree-based approach is much easier to implement and provides to some extent shorter run times, its routing quality is similar to the Steiner-tree-based approach.

9. Conclusion

We have proposed two new methodologies for current-driven routing of non-planar signal nets. These approaches are to our knowledge the first routing algorithms capable of constructing multi-terminal signal nets with current-correct wire widths without the need for a separate layout post-processing step.

Both methods have been successfully used to generate current-correct designs of "real world" circuits. Currently, both approaches are being integrated into commercial design flows of analog circuits for automotive applications where further tests and comparisons will be performed. As future work, we are optimistic that the presented methodologies may be useful in addressing electromigration in deep-sub-micron designs of digital circuits, and plan to investigate this further.

10. References

[1] J. R. Black, "Electromigration Failure Models in Aluminum Metallization for Semiconductor Devices," *Proc. of the IEEE*, Vol. 57, no. 9, 1969, pp. 1587-1594.

[2] J. R. Black, "Physics of Electromigration," *Proc. IEEE Int. Reliability Physics Symposium*, 1983, pp. 142-149.

[3] S. Chowdhury, "An Automated Design of Minimum-Area IC Power/Ground Nets," *Proc. Design Automation Conf.*, 1987, pp. 223-229.

[4] F. M. D'Heurle, "Electromigration and Failure in Electronics: An Introduction," *Proc. of the IEEE*, Vol. 59, no. 10, 1971, pp. 1409-1417.

[5] M. R. Garey, D. S. Johnson, "The Rectilinear Steiner Tree Problem is NP-Complete," *SIAM Journal Applied Math*., 1977, pp. 826-834.

[6] F. O. Hadlock, "The Shortest Path Algorithm for Grid Graphs," *Networks*, vol. 7, 1977, pp. 323-334.

[7] S. Haruyama, D. Fussel, "A New Area-Efficient Power Routing Algorithm for VLSI Layout,", *Proc. ICCAD*, 1987, pp. 38-41.

[8] C. Y. Lee, "An Algorithm for Patch Connections and Its Applications," *IRE Trans. Electron. Comput*., vol. EC-10, 1961, pp. 364-365.

[9] J. H. Lee, N. K. Bose, F. K. Hwang, "Use of Steiner's Problem in Suboptimal Routing in Rectilinear Metric," *IEEE TCAS*, vol. CAS-23, 1976, pp. 470-476.

[10] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*, John Wiley & Sons, 1990, pp. 405-406.

[11] T. Mitsuhashi, E. S. Kuh, "Power and Ground Network Topology Optimization," *Proc. Design Automation Conf.,* 1992, pp. 524-529.

[12] A. S. Moulton, "Laying the Power and Ground Wires on a VLSI Chip," *Proc. Design Automation Conf.,* 1983, pp. 754-755.

[13] H.-J. Rothermel, D. A. Mlynski, "Automatic Variable-Width Routing for VLSI," *IEEE TCAD*, vol. CAD-2, no. 4, Oct. 1983, pp. 271-284.

[14] R. A. Rutenbar, J. M. Cohn, "Layout Tools for Analog ICs and Mixed-Signal SoCs: A Survey," *Proc. of the Int. Symposium on Physical Design*, 2000, pp. 76-83.

[15] Z. A. Syed, A. Gamal, "Single Layer Routing of Power and Ground Networks in Integrated Circuits," *Journal of Digital Systems*, vol. VI, no. 1, 1982, pp. 53-63.

[16] D. Young, A. Christou, "Failure Mechanism Models for Electromigration," *IEEE Trans. on Reliability*, Vol. 43, no. 2, 1994, pp. 186-192.

[17] S. Q. Zheng, J. S. Lim, S. S. Iyengar, "Finding Obstacle-Avoiding Shortest Paths Using Implicit Connection Graphs," *IEEE TCAD*, vol. CAD-15, no. 1, Jan. 1996, pp. 103-110.

[18] Patents pending US 5736426 and US 5888893.