# Single Step Current Driven Routing of Multiterminal Signal Nets for Analog Applications[*]

Thorsten Adler[†]
Infineon Technologies AG
Thorsten.Adler@infineon.com

Erich Barke
Institute of Microelectronic Systems
University of Hanover, Germany
Erich.Barke@ims.uni-hannover.de

## Abstract

*We present the single layer router CDR (Current Driven Router) capable of routing analog multiterminal signal nets with current driven wire widths. The widths used during routing are determined by current properties per terminal gained by simulation or manually specified by circuit designers.*

*The algorithm presented computes a Steiner tree layout satisfying all specified current constraints while obeying the maximum allowed current densities on all connections. CDR calculates the Steiner tree topology, computes the unknown currents of wires connecting two Steiner points and generates the final Steiner tree layout in a single step thus eliminating the need for a separate layout post-processing step common to power and ground routing algorithms.*

*CDR uses a connection graph for layout representation and applies an advanced minimum detour algorithm in combination with a modified 'three-point steinerization' heuristic for Steiner tree based layout construction.*

## 1. Introduction

Designing analog power ASICs in modern mixed signal BCD-processes (Bipolar, CMOS, DMOS) requires a large amount of expert knowledge in order to meet constraints like symmetry, voltage drops, current density, temperature gradients, piezoelectrical effects, electromigration, etc. Unlike in digital design it is not possible to treat all of these constraints automatically up to now.

One main difference with respect to signals in digital circuits is the presence of large currents. In order to avoid electromigration due to excessive current density one has to design wires according to the current imposed on them. Routing multiterminal nets with current driven wire widths is a problem which arises not only in power and ground routing of analog and digital circuits but also in routing of multiterminal signal nets in analog circuits.

Several approaches to automatic routing of power and ground nets have been presented ([2], [19], [20], [25]). Their goal is to generate the power supply interconnect of integrated circuits prior to 'normal' routing in order to achieve a planar single-layer implementation of these nets. Routing of power and ground nets consists of three tasks: Construction of interconnection topology, wire width determination and layout generation. [19] computes the power and ground net topology using a combination of Hightower's line-search algorithm [9] and Lee's maze-routing algorithm [12] using a standard wire width. Based on that topology all unknown currents, i.e. the currents of wires connecting Steiner points[1], are calculated. Afterwards, all wires are widened with respect to current flow. This may lead to DRC errors that have to be resolved in a separate post-processing step which modifies device placement.

In this paper we present a new approach to current driven routing of multiterminal nets for analog circuits. Unlike in power and ground routing algorithms our algorithm calculates the unknown currents 'on the fly' during Steiner tree based layout construction. Therefore, no post-processing steps are needed to generate design rule correct layout. In order to achieve good routing results even in congestioned layout regions CDR uses more advanced detailed routing algorithms than those used by power and ground routers normally working on an empty routing space.

In the next two sections we describe basic path finding algorithms and algorithms for routing multiterminal nets. Section 4 illustrates CDR's database and basic routing algorithm and in Section 5 we present the algorithm used to route a multiterminal net with current driven wire widths. Section 6 illustrates some examples generated with CDR and finally we give some concluding remarks.

[*] This work was supported by the German BMFT under contract 01 M 3034. The authors are responsible for the contents of this publication.

[†] This work was done while T. Adler was with IMS Hanover

[1] Assuming that the currents at all terminal are known

## 2. Path Finding Algorithms

There are two basic classes of shortest path finding algorithms: Maze-routing algorithms and line-searching algorithms. Maze-routing algorithms rely on a grid-based layout representation. The first known algorithm is Lee's algorithm [7] which uses an improved version of the breadth first search (BFS) to find the shortest path between source *s* and target *t*. It requires *O(mn)* memory and run-time in the worst case for *m x n* grid graphs.

In recent history there have been numerous improvements (e.g. [1], [6], [24]) to reduce memory and run-time requirements. Hadlock presented the Minimum Detour (MD) algorithm [6] which uses the *A** search heuristic described in [8]. The MD algorithm is a shortest path algorithm which is controlled by a parameter, called detour number, leading to a drastically reduced search space.

All partial paths generated by maze routing algorithms are represented by a sequence of grid points with a width of the grid size *g*. In order to generate a route with a width *w* different from the grid size *g* one has to modify the basic maze-routing algorithm in such a way that a set of grid points representing the desired wire width is labeled at once. This increases run-time even further.

To improve performance over maze-routing algorithms, line-search algorithms have been proposed. Line-search algorithms tend to reduce memory requirements by using line segments instead of grid nodes during path searching. The time and space complexity of these algorithms is *O(L)*, where *L* is the number of line segments produced. The first line-search algorithms where reported in [9] and [15] almost simultaneously. While [9] generates fewer trial lines during path searching and thus achieves a faster algorithm, [15] generates trial lines at every grid point and thus guarantees the optimal solution. In addition, routing with different wire widths is much easier as it only requires expanding trial rectangles instead of trial lines.

More recent line search algorithms (e.g. [4], [16], [18], [26]) are based on computational geometry techniques. Almost all of these algorithms rely on a graph that is more sparse than the original grid. Such a graph is called a *connection graph* in [14].

## 3. Routing Multiterminal Nets

The problem of finding the shortest connection for a net with *n* terminals ($n \geq 3$) is called the Steiner problem [11]. It is related to the spanning tree problem. However, in addition to the *n* terminals, it uses further connection points called *Steiner points*.

Hanan [7] showed that there always exists an MRST (Minimum Rectilinear Steiner Tree) with Steiner points chosen from the intersections of all horizontal and vertical lines passing through all terminals of the net. Despite this restricted solution space, the MRST problem remains NP-complete [5]. This has given rise to numerous heuristics as surveyed by [11], [21], [22], [23]. Most heuristics start with a minimum spanning tree (MST) and modify it to obtain shorter trees, because it has been shown that the length of an MST is at most 3/2 of the length of the MRST [10]. Thus, an MST is a good approximation for an MRST.

Lee, Bose and Hwang [13] proposed a variation of Prim's algorithm [17] called 'Three-Point Steinerization' (P3S). P3S generates an MRST by sequentially adding the nearest terminal to the current subtree. That terminal is chosen based on the minimum Manhattan distance to a terminal or Steiner point in the current subtree. The connection between those three points is built using three-point RMSTs (Rectilinear Minimum Spanning Tree).

## 4. CDR's Database and Basic Algorithms

CDR's database and path searching algorithms are based on those presented in [27]. A connection graph $G_C$ is used for layout representation. It can be obtained by extending the horizontal and vertical edges of each obstacle until another obstacle or the boundary is reached, in addition to generating a horizontal and vertical line through all terminals (see Fig. 1). For a more formal definition of $G_C$ refer to [27] or [14].

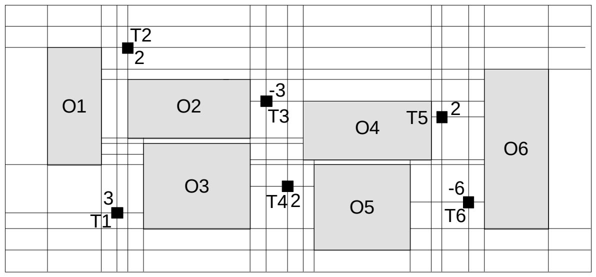


**Fig. 1: An example connection graph $G_C$**

It is known that the shortest path between source and target is a path in the connection graph [14], [27]. Therefore, a shortest path algorithm such as Dijkstra's algorithm [3] can be employed to find a path of minimum length between source and target.

The algorithm *detour()* used to accomplish this is similar to the one presented in [27]. It is basically a line-search version of the MD algorithm with a generalized detour number concept. The detour length of a node *u* with respect to a source node *s* and a target node *t*, denoted by *d(u)*, is the sum of the detour lengths of all directed edges in any directed shortest path from *s* to *u* in $G_C$.

## 5. CDR’s Steiner Tree Construction

Due to the more generalized problem of Steiner tree based layout construction with non-uniform wire widths CDR has to cope with additional difficulties with respect to basic Steiner tree algorithms.

One problem for the current driven router is the determination of realistic current values for each terminal. These current values are determined using a standard circuit simulator or manually specified by analog circuit designers. A post processing step is used after simulation to extract a set of ‘worst case’ current vectors each representing a snapshot of the circuits operation at a particular point of time.

Figure 2 illustrates an example net with six terminals using only one current value per terminal for simplicity[2]. The net shown has two current sources (terminals T3 and T6) and four current sinks (T1, T2, T4, T5) which is indicated by the positive current value shown. A ‘standard’ Steiner tree algorithm using a uniform (i.e. minimum) wire width would lead to the net topology shown in Figure 2 (Steiner points ST1, ST2, ST3, ST4):

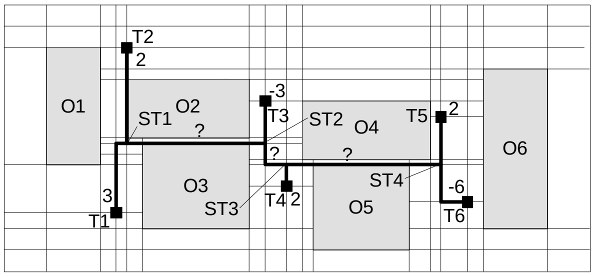


**Fig. 2: An example net topology**

The current flow on wires connecting two Steiner points (e.g. ST1 and ST2) 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 2 obstacles O2/O3 and O4/O5 would have to be moved in order to generate the final layout.

Therefore, CDR’s Steiner tree algorithm has to build the Steiner tree in a greedy, sophisticated fashion to compute the unknown wire widths ‘on the fly’ during Steiner tree construction. CDR’s Steiner tree construction is based on a modification of the P3S algorithm [13] described in Section 3. The P3S algorithm sequentially adds the nearest terminal to the current subtree. That terminal is determined using simple Manhattan distances. Due to the presence of obstacles CDR’s algorithm has to use a smarter method to add a terminal to the partially routed subtree.

[2] These set of current values represents a snapshot of the circuit behavior at a particular point of time

```
CDR()
 {
   sort 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 calculates the Steiner tree layout by repeatedly computing an optimum Steiner point for three terminals at a time. At first, *calc_steiner()* computes the optimum Steiner point for the first three terminals. 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()*.

Using this greedy Steiner tree construction CDR is able to compute the unknown current flow on connections between two Steiner points by simply adding the current flows of the two wires connecting to the Steiner point as shown in Figure 3:

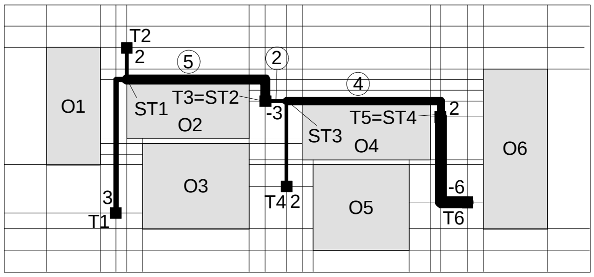


**Fig. 3: A smarter solution**

At Steiner point ST1 CDR has to add the current flows of terminal T1 and terminal T2 to compute the unknown current on the wire which leaves Steiner point ST1. At Steiner points ST2, ST3 the current flow of terminals T3, T4 is added to this sum and finally the current flow of terminal T5 is added to that value at Steiner point ST4.

*Calc_steiner()* computes the optimum Steiner point for a given set of three terminals at a time and uses *detour()* to compute the exact distances between the candidate Steiner point *u* and the two target terminals *t1* and *t2*. C*alc_steiner()* is guided by a cost function which uses the priority queue *PQ* to store the candidate Steiner points and basically computes the resulting routing area for each candidate Steiner point. These points are reached using a breadth first search starting from the source terminal.

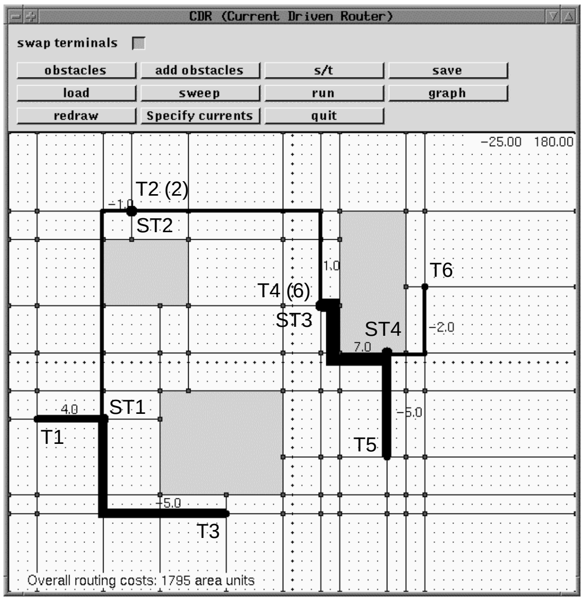
```
calc_steiner(source, t1, t2, width[source], width[t1], width[t2])
{
  width:=width[source]; PQ:=∅; dl[source]:=0; steiner:=source;
  u:=source
  d1:=detour(u, t1, width[t1]); d2:=detour(u, t2, width[t2])
  if (d2<d1)
      swap_terminals(t1, t2)
  cost(u):=  width[t1]*detour(u, t1, width[t1])+
             width[t2]*detour(u, t2, width[t2])+
             estimate_route_area(u)
  min_cost:=cost(u); PQ.insert(u, cost(u))
  while (!PQ.empty())
      {
      u:=PQ.delete_min(); visited.insert(u); d:=dl[u]
      for each neighbor v of u in G_C and v∉visited
             {
             cost(v):=  width[source]*(d+M(u, v))+
                        width[t1]*detour(v, t1, width[t1])+
                        width[t2]*detour(v, t2, width(t2))+
                        estimate_route_area(v)
             if (cost(v)<min_cost)
                 PQ.insert(v, cost(v)); min_cost:=cost(v);
                 steiner:=v;
             dl[v]:=dl[u]+M[u, v]
             }
      }
  return (steiner)
 }
```

At first, *calc_steiner()* computes the cost function for a candidate Steiner point at the source node and stores that value into the priority queue *PQ*. Afterwards all reachable neighbors of the current node are evaluated. They are stored into *PQ* if their cost is lower than the current minimum cost. This continues until *PQ* is empty which means that no further cost reduction (i.e. routing area reduction) was possible. The node last entered into *PQ* is the one which minimizes the cost function and therefore the optimum Steiner point for the given set of terminals.

In order to find the global minimum *calc_steiner()* uses some heuristics to improve the monotone search algorithm. If, for example, the distance between the first candidate Steiner point (*source*) and the second target terminal *t2* is smaller than the distance between *source* and terminal *t1*, *t1* and *t2* will be swapped in order to improve the routing result.

## 6. Examples

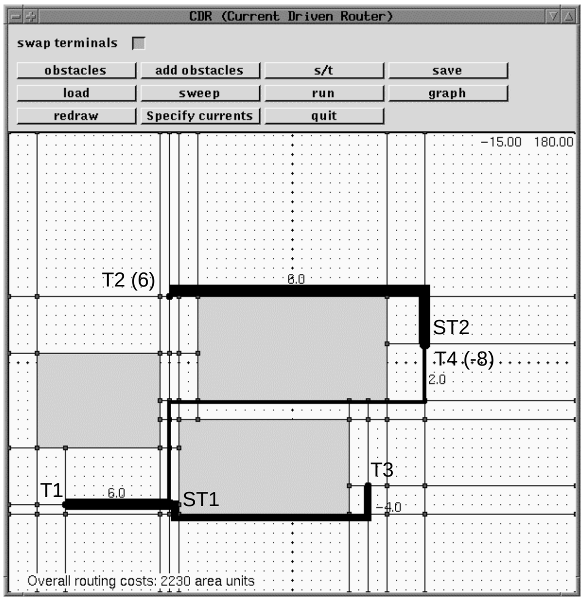
Figure 4 illustrates CDR's result for an example net with 6 terminals. To obtain better results terminals T2 and T3 were swapped during calculation of Steiner point ST1. Furthermore, Steiner points ST2 and ST3 were placed above terminals T2 and T4, respectively.



Overall routing costs: 1795 area units

**Fig. 4: An example net with 6 terminals**

The example shown in Figure 5 was routed without any necessary layout modification due to CDR's enhanced Steiner tree generation:



Overall routing costs: 2230 area units

**Fig. 5: A second example with 4 terminals**

CDR swapped terminal T2 and terminal T3 and placed Steiner point ST2 above terminal T4. A standard power and ground routing algorithm would connect terminal T1 and terminal T2 which would lead to an resulting routing width of 12 units. In order to create a design rule correct layout one obstacle would have to be moved after topology determination which is not applicable for analog design.

## 7. Conclusion

In our paper we presented a new approach to current driven routing of multiterminal nets for analog circuits, called CDR. The current properties used to guide routing are gained by simulation or manually specified by analog circuit designers. CDR generates Steiner tree based layout in a greedy fashion in order to compute the unknown wire widths between two Steiner points ‘on the fly’

The algorithm used for point-to-point connections is based on a modification of the MD algorithm similar to [27] and operates on a connection graph used for layout representation. CDR’s Steiner tree algorithm is a variation of the P3S algorithm [13] which uses Prim’s algorithm [17] to compute a Steiner tree by sequentially adding the nearest terminal to the already routed subtree. The cost function used for Steiner point determination is based on a routing area estimation which calculates the exact distances between the candidate Steiner point and the next two terminals and estimates the routing area of the remaining net. The Steiner tree algorithm incorporates some heuristics such as terminal swapping during routing to improve the routing result.

CDR is to our knowledge the first routing algorithm capable of constructing Steiner tree based layouts of multiterminal signal nets with current driven wire widths in a single step, thus eliminating the need of a separate layout post-processing step. It is currently being integrated into two commercial design flows and will be used to enhance automatic layout generation for analog integrated circuits.

## References

[1] S. B. Akers, *A modification of Lee’s path connection algorithm*, IEEE Trans. Electron. Comput., vol. EC-16, pp. 97-98, 1967

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

[3] E. W. Dijkstra, *A Note on two problems in connexion with graphs*, Numer. Math., vol. 1, pp. 269-271, 1959

[4] K. L. Clarkson, S. Kapoor and P. M. Vaidya, *Rectilinear shortest paths through polygonal obstacles in $O(n(\log n)^2)$ time*, Proc. Third Annual Conf. Computational Geometry, pp. 251-57, 1987

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

[6] F. O. Hadlock, *The shortest: Path algorithm for grid graphs*, Networks, vol. 7, pp. 323-34, 1977

[7] M. Hanan, *On Steiner's Problem With Rectilinear Distance*, SIAM Journal of Applied Mathematics, vol. 30, no.1, pp. 324-342, April 1972

[8] P. Hart, N. Nilson and B. Raphael, *A formal basis for the heuristic determination of minimum cost paths*, IEEE Trans. Syst., Sci., Cybern., vol. SCC-4, pp. 100-107, 1968

[9] D. W. Hightower, *A Solution to line routing problems on the continuous plane*, DA Workshop, pp. 1-24, 1969

[10] F. K. Hwang, *On Steiner Minimal Trees with Rectilinear Distance*, SIAM Journal on Applied Math., vol. 30(1), pp. 104-114, 1976

[11] F. K. Hwang, D. S. Richards and P. Winter, *The Steiner Tree Problem*, Annals of Discrete Mathematics, No. 53, North-Holland, 1992

[12] C. Y. Lee, *An Algorithm for Patch Connections and Its Applications*, IRE Trans. Electron. Comput., vol. EC-10, pp. 364-65, 1961

[13] J. H. Lee, N. K. Bose and F. K. Hwang, *Use of Steiner’s problem in suboptimal routing in rectilinear metric*, IEEE TCAS, vol. CAS-23, pp. 470-476, 1976

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

[15] K. Mikami and K. Tabuchi, *A computer program for optimal routing of printed circuit connectors*, Proc. IFIPS, vol. H-47, pp. 1475-78, 1968

[16] J. S. B. Mitchell, *An optimal algorithm for shortest rectilinear paths among obstacles in the plane*, Abstracts First Canadian Conf. Comput. Geometry, pp. 22, 1989

[17] R. C. Prim, *Shortest connection networks and some generalizations*, Bell System Technology Journal, vol. 36, pp. 1389-1401, 1957

[18] P. J. Rezend, D. T. Lee and Y.-F. Wu, *Rectilinear shortest paths with rectangular barriers*, Proc. Second Annual Conf. Computational Geometry, pp. 204-13, 1985

[19] H.-J. Rothermel and D. A. Mlynski, *Computation of Power Supply Nets in VLSI Layout*, Proc. Design Automation Conference, pp. 37-47, 1981

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

[21] M. Sarrafzadeh and C. K. Wong, *An Introduction To VLSI Physical Design*, McGraw-Hill, pp. 96-107, 1996

[22] M. Servit, *Heuristic algorithms for rectilinear Steiner trees*, Digital Processes, vol. 7, pp. 21-32, 1981

[23] N. A. Sherwani, *Algorithms for VLSI Physical Design Automation*, Kluwer Academic Publ., pp. 228-36, 1993

[24] J. Soukup, *Fast maze router*, Proc. Design Automation Conference, pp. 100-102, 1978

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

[26] Y.-F. Wu, P. Widmayer, M. D. F. Schlag and C. K. Wong, *Rectilinear shortest paths and minimum spanning trees in the presence of rectilinear obstacles*, IEEE Trans. Comput., vol. C-36, pp. 321-31, 1987

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