An Interactive Router for Analog IC Design[*]

Thorsten Adler
Institute of Microelectronic Systems
University of Hanover, Germany
adler@ims.uni-hannover.de

Jürgen Scheible
Robert Bosch GmbH
72703 Reutlingen, Germany

Abstract

*We present an interactive two layer router integrated in an analog IC design environment used in an SDL (**s**chematic **d**riven **l**ayout) design flow. Special features are its customizability, the treatment of arbitrary polygons and an advanced handling of source/target polygons in order to avoid net internal design rule violations during connection phase.*

A global routing algorithm is used to split the route into separate parts each routable in a single layer. After via placement a specialized maze router performs the advanced single layer routes in 90 or 45 degree mode. The resulting route can be modified by interactive via movement and rerouting of obsolete partial routes.

1 Introduction

Designing analog 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 a.s.o.. Unlike in digital design it is not possible to treat all of these constraints automatically up to now.

Several approaches to automatic synthesis and layout of analog cells have been presented ([DeG87], [Rij89], [Koh90], [Car89], [Gie90], [MzB93]). Their goal is to generate the final layout of an analog cell following a given set of specifications without the need of an expert analog designer. In such cases these tools can be very efficient and save considerable design time. However, most of these systems are devoted to one or several specific classes of analog cells and cannot easily be extended to other classes.

To overcome these shortcomings an interactive layout style as reported in [DoDu91] seems to be the right choice. Today, the SDL design flow is widely used for analog and mixed signal ASIC design. The power of this design flow builds on using automated algorithms in an interactive design environment to speed up IC design.

The SDL design flow used at R. Bosch GmbH [Sch96] is based on Mentor Graphics' ICstation and uses customized device generators to generate analog cells manually placed by expert analog designers. For the routing process a routing tool has been developed which consists of two routing algorithms, ANALOGROUTER (AR) and GLOBALROUTER (GR).

This routing tool was not designed to perform automatic routing of complete circuits but to provide the circuit designer with an interactive tool capable of improving designer's efficiency. It contains special features required by Bosch's analog designers.

In the following section we describe the single layer maze routing algorithm AR, developed at R. Bosch GmbH. In Section 3, we present the global routing algorithm GR, developed at IMS, and in Section 4 we give a brief overview of the implementation and integration into the Mentor Graphics ICstation environment. An example layout done with GR and AR will be shown. Finally, we give some concluding remarks.

2 ANALOGROUTER

Within Mentor Graphics' ICstation all multi-port nets are decomposed into series of symbolic two point connections between already layouted net components. Therefore, AR was built to create a connection between two objects, source (S) and target (T). S and T can be ports of a device or elements of a partly routed net, arbitrary polygons as well as paths. When multiple two point connections are selected for routing, AR performs a heuristic net sorting to bring the connections into an order which is favorable to the routability.

[*] This work was supported by german BMBF under JESSI AC12 contract No. 01M2885K

2.1 Basic principle and database

AR is a single layer maze router based on the well known Lee algorithm [Lee61] and shows the following special features:

- 45 degree option for HVD routing (horizontal, vertical and diagonal path segments),
- layer independent routing directions combined with routing over active area,
- special distance rules in diagonal directions and current driven wire widths,
- treatment of arbitrary polygons with special connecting behaviour.

Wire width *w* can be chosen by the user or is determined by the current flowing through S and T specified by properties in the schematic. AR's database is a two-dimensional bitmap, which represents a rectangular part of the current layout, called 'routing area'.

The algorithm consists of three steps: Database generation through oversizing due to distance rules (Sect. 2.2), path searching algorithm (Sect. 2.3) and connection behaviour (Sect.2.4).

2.2 Oversizing

The routing algorithm is based on a modification of the well known Lee algorithm which calculates paths of one grid width only. Therefore, for each route all polygons have to be oversized by $s = d_{\min} + w/2$ where $d_{\min}$ is the minimum distance between two wires and *w* is the current wire width. This operation guarantees, that the calculated one grid path can be replaced by a path of width *w* without violating the design rules. Figure 1 shows different possibilities for sizing. In a true sizing procedure circle segments originate from corners (Fig. 1a). For HV routing a sizing procedure which just moves edges is adequate (Fig. 1b). For HVD routing a sizing procedure which squashes 90 degree corners to diagonal ones is needed to enable the path searching algorithm to create diagonal path segments (Fig. 1c). This is important in order to avoid electromigration, because current density at diagonal corners is two times less than that in 90 degree corners.

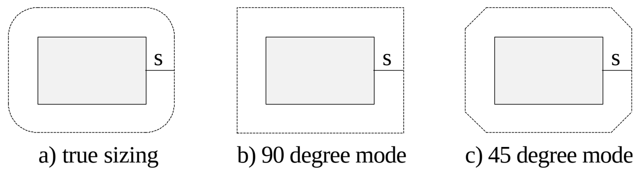


Figure 1: Different oversizing strategies

2.2.1 Special distance rules in HVD-routing

Modern mixed signal processes like BCD exhibit direction dependent distance rules. A special rule of this kind prescribes a larger spacing between metal wires in diagonal directions. This rule requires a special sizing procedure in diagonal directions. Another rule demands that all coordinates have to be on grid. This leads to the requirement that the diagonal path segments must be widened in order to bring the corners of the wire shape on grid.

Both rules can be fulfilled by using direction dependent sizing values s_{45} and s_{90} where $d_{45} > d_{90}$ and $w_{45} > w_{90}$ are the direction dependent distances and wire widths, respectively, as shown in Figure 2.

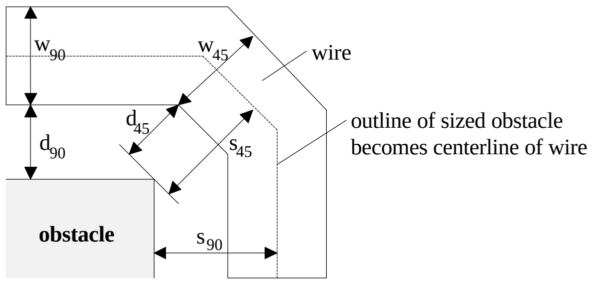


Figure 2: Special design rules in diagonal directions

To provide this special oversizing a new sizing method has been implemented in AR.

2.2.2 Direction dependent sizing

The sizing algorithm works on databases in which polygons are represented by their corners in anti clockwise order. Following this order each corner is replaced by one or more corners of an octagon, whose center is placed on the original corner (Fig. 3).

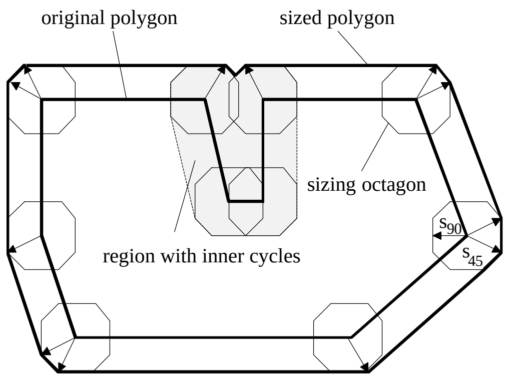


Figure 3: Oversizing with direction dependent sizing values using sizing octagons

The number of corners created by one corner of the original polygon (see arrows in Fig. 3) is determined by the angles of the corresponding edges. The octagon is given by the sizing values s_{90} and s_{45}.

During sizing it is checked if a new edge intersects with one already created. This case indicates an inner cycle which is eliminated immediately (see Fig. 3). Holes within the original polygon are also deleted as inner cycles which does not affect routing behaviour.

Standard sizing algorithms with one unique oversizing value for all directions have to use s_{45} for sizing in order to fulfill all distance rules. This leads to a waste of space in orthogonal directions because $s_{90} < s_{45}$. Our sizing method guarantees that all distance rules are kept without wasting space and has been applied for patent [DE7].

2.3 Path searching algorithm

The distance D between two points $(x1, y1)$ and $(x2, y2)$ is defined as $D = \sqrt[n]{|x_2 - x_1|^n + |y_2 - y_1|^n}$. In Lee‘s wave the distance of a bitmap element to the source S, where the wave has started, is given in manhattan metric ($n = 1$): Orthogonal neighbours in the bitmap have a distance $D_1 = 1$, diagonal neighbours distance $D_1 = 2$. In this case, the set of bitmap points with the same index *i* has the shape of a square, if S is a single point. The HVD-wave of AR must propagate in euclidean metric ($n = 2$) in order to define real distances. Therefore, diagonal neighbours should have the distance $D_2 = \sqrt{2}$. This is approximated by the value $D_{HVD} = 1.5$. In order to propagate the wave with integer values orthogonal neighbours of an element *i* are indicated with *i+2*, diagonal neighbours with *i+3*. This leads to wave fronts nearly shaped as circles.

A 'tunnel polygon' calculated by GR (see Section 3) is used to restrict wave propagation during path searching. Direction preserving backtracking from target to source and path generation is similar to Lee's algorithm.

2.4 Special connecting behaviour

When S or T are arbitrary polygons net internal design rule violations may occur in the region, where the generated wire connects to the polygon. Design rule violations of this kind can be angles less than π/2 or distance rule violations between *opposite*[1] edges (Fig. 4).

[1] Edges are opposite, if their prolongation’s include an angle less than π/2

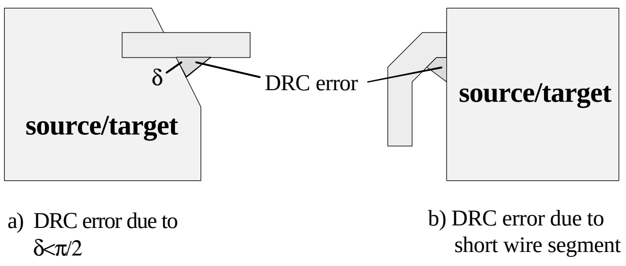

δ
DRC error
source/target
source/target
a) DRC error due to δ<π/2
b) DRC error due to short wire segment

Figure 4: Net internal design rule violations

Design rule violations of type (b) are avoided by the definition of minimum lengths for the end segments of a connecting wire. This can be achieved by oversizing S and T before path searching and prolongation of the generated path afterwards. Design rule violations of type (a) are avoided by defining *connectable* points on the edges of S and T.

In Figure 5, connectable points on source and target are indicated by points and thick lines. Allowed directions for end segments of wires are indicated by arrows. The wire connecting to S or T must end on a connectable point with an allowed direction.

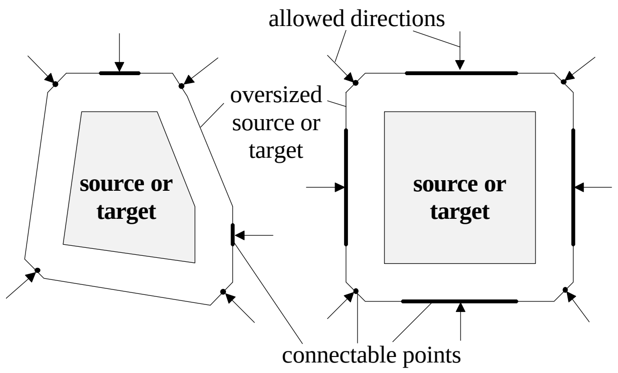


Figure 5: Connecting to source or target

The methods described have been applied for patent [DE0]. AR has been purchased by Mentor Graphics and is now commercially available to all users.

3. GLOBALROUTER

GR is a two layer global routing algorithm for two point connections and was developed as an extension to AR adding new features. As with AR source and target can be ports of a device or elements of a partly routed net, arbitrary polygons as well as paths. To speed up routing GR runs on a coarse grid (see Sect. 3.2) and computes so called 'tunnel polygons' which are used by AR to restrict wave propagation during detailed single layer routing. GR has the following features in addition to those mentioned in Section 2: two layer routing, different wire widths for each route and each layer and different routing costs for each routing layer and vias.

3.1 Basic principle and database

GR splits the overall two layer route into separate parts each routable in a single layer. It calculates via positions and calls AR to perform the detailed single layer routes between S, T and the computed vias.

GR's database is a two-dimensional bitmap which represents a rectangular area of the current layout. Each data element represents one grid point. The routing algorithm is based on AR's algorithm and, therefore, all polygons have to be oversized by $s = d_{min} + w/2$ (see Section 2.2).

Each grid point contains a 32 bit integer value which holds information about the current layout situation. Bits 31 and 30 are used to mark all grid points inside source or target polygons. Bits 21 to 29 store information whether it's forbidden to extend the wave from the current grid point to the neighbouring grid points. The least significant 22 bits are used to store the index during wave propagation (see Sect. 3.2).

Figure 6 shows a polygon and its representation inside GR's database.

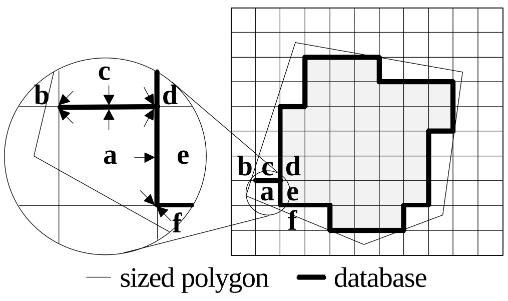


Figure 6: Layout representation inside GR's database

The arrows in the enlarged grid point *a* in Figure 6 mean that it is forbidden to propagate the wave from grid point *a* into grid points *b*, *c*, *d*, *e* and *f*. For each forbidden direction a flag is set in the corresponding integer value. For source and target polygons each grid point touched by the polygon is marked as source or target. The source grid points are stored in the initial wavefront (see Sect. 3.2).

To reduce memory requirements and runtime the grid size is chosen as large as possible. After sizing the smallest possible polygon has a width of $W_{\min} = w_{\min} + 2s$ which equals to $W_{\min} = 2w_{\min} + 2d_{\min}$ for wires with minimal width. In order to store all polygons correctly the grid size *G* has to be: $G < W_{\min}$.

Furthermore, all sized polygons must not overlap. Figure 7 gives an example for two overlapping polygons. After sizing all polygons are stored sequentially in the internal database. Due to the layout representation used it seems possible to propagate the wave from grid point *a* to grid point *b*. Therefore a boolean OR is performed in order to generate non-overlapping polygons (Fig. 7).

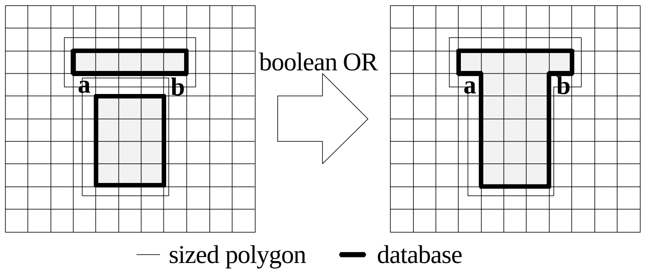


Figure 7: Two overlapping polygons

GR's database consists of three levels containing source, target and obstacle polygons: The two routing layers and an intermediate layer used to calculate the via positions during global routing. This intermediate layer is a boolean OR of layer 1 obstacles oversized by s_{via1}[2] and layer 2 obstacles oversized by s_{via2} yielding all space not usable for placement of via origins.

3.2 Wave propagation

GR's wave propagation can be customized by the cost function $f = l_1 \cdot cost_{layer1} + l_2 \cdot cost_{layer2} + num\_of\_vias \cdot cost_{via}$ with $cost_{layer1}$, $cost_{layer2}$ and $cost_{via}$ representing the cost factors for routes in layer 1, routes in layer 2 and vias, respectively. The wave propagation of GR is similar to that of AR. In HV and HVD mode all bitmap elements *i* (elements with index *i*) label their applicable[3] orthogonal neighbours $n_O(i)$ with $i + 2 \cdot cost_{layer}$. The upper or lower neighbour in the other routing layer is labeled with $i + 2 \cdot cost_{via}$. In HVD mode this step is followed by marking all applicable diagonal neighbours $n_d(i)$ with $i + 3 \cdot cost_{layer}$. This sequential labeling guarantees that no grid element is labeled twice and, therefore, reduces runtime [Oht86]. All newly labeled grid points are stored in a priority queue which consists of a number of linear lists. A single linear list is used to store one single wavefront[4]. Therefore, no sorting is needed.

[2] $s_{via1} = d_{layer1} + via\_width_{layer1}/2$

[3] Neighbours which are not yet indicated and could be reached from element *i*

[4] A wavefront contains all grid elements with the same index

3.3 Path generation

After the wave has reached the target polygon the path from source to target with minimum costs can be determined. This is done similarly to the algorithm used by AR. The resulting path contains all grid points (grid size *G*) in which the detailed route done by AR should be realized. GR converts the resulting path into a tunnel polygon and calls AR in order to perform the final route. This polygon is used by AR to restrict the wave propagation to grid points inside that polygon. GR calculates the exact via positions and calls AR for every single layer route needed.

3.4 Program flow

GR's program flow is shown in Figure 8:

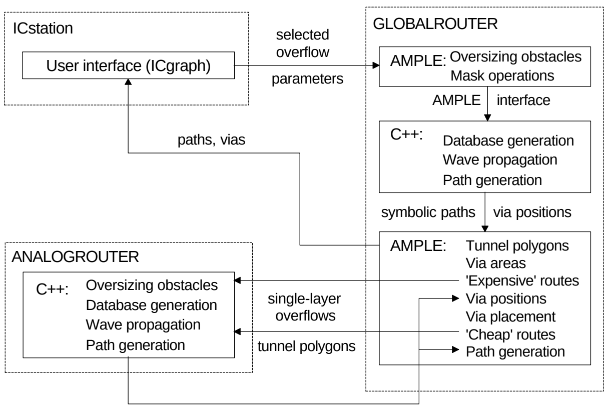

ICstation
User interface (ICgraph)
selected overflow parameters
GLOBALROUTER
AMPLE: Oversizing obstacles Mask operations
AMPLE interface
C++: Database generation Wave propagation Path generation
paths, vias
symbolic paths
via positions
AMPLE: Tunnel polygons Via areas 'Expensive' routes Via positions Via placement 'Cheap' routes Path generation
ANALOGROUTER
C++: Oversizing obstacles Database generation Wave propagation Path generation
single-layer overflows
tunnel polygons

Figure 8: GR's program flow

At first, the database is generated through polygon sizing and boolean mask operations. Afterwards, GR computes the global route. This step yields the tunnel polygons and via positions in coarse grid *G*. Figure 9 shows an example for $cost_{layer1} < cost_{layer2}$.

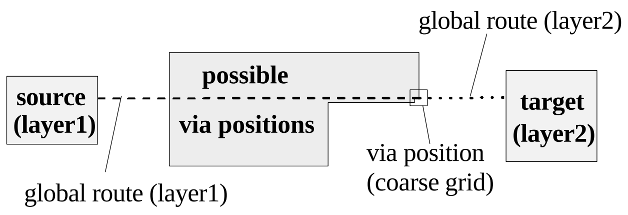


Figure 9: The global route

In order to shorten the 'expensive' routes AR is called to route these connections first although the exact via positions (grid *g*) are not yet known. Therefore, the via holes are used as temporary targets for these routes. These holes are computed by a boolean AND between the via positions in coarse grid and all possible via positions calculated during database preparation.

Afterwards, GR computes the exact via positions which are determined by the start and/or end points of the just performed 'expensive' routes (see Fig. 10). After via placement all 'cheap' routes are done by AR to complete routing.

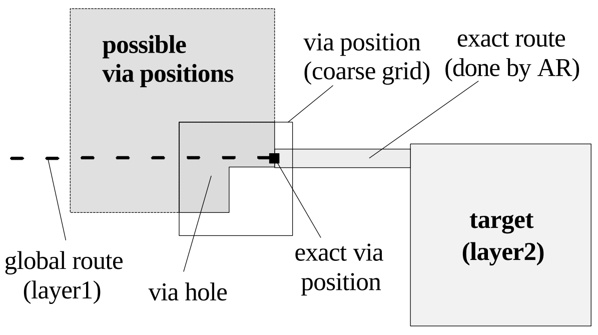


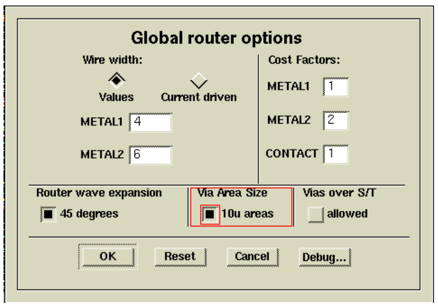
Figure 10: Determining exact via positions

4. Implementation and example

The algorithms described were integrated into Mentor Graphics' ICstation environment to speed up analog IC design at R. Bosch GmbH. The algorithms were implemented using C++ and Mentor Graphics' build-in programming language AMPLE (see Fig. 8). AMPLE is mainly used to provide the user interface between AR, GR and the ICstation. AR is totally written in C++ whereby AMPLE is used in GR to perform polygon sizing, boolean mask operations and via calculations. In a later version of GR all routines will be implemented in C++ to speed up routing.

To provide greater flexibility and greater interaction the designer has the ability to move the placed vias inside their via holes if he is not satisfied with automatic via placement (see Fig. 10, 12 and 13). After via movement a reroute function is used to reroute the now obsolete routes connecting to the moved via(s).

All parameters like routing costs and wire widths are customizable through an interface shown in Figure 11.


Global router options

Wire width: [✓ Values] Current driven

METAL1	4
METAL2	6

Cost Factors:

METAL1	1
METAL2	2
CONTACT	1

Router wave expansion: [✓ 45 degrees]

Via Area Size: [✓ 10u areas]

Vias over S/T: allowed

Figure 11: GR's user interface

The route depicted in Figure 12 was accomplished in about 3 seconds CPU time on a SUN SPARCstation 5 with routing costs set to [1,2,1] (see Fig. 11). This includes five calls to AR to perform the detailed single layer routes. Figure 13 shows the same example with routing costs set to [2,1,1]. For this route placement of vias above source or target polygons was enabled.

5. Conclusion

In our paper, we presented two routing algorithms AR and GR developed at IMS Hanover and R. Bosch GmbH. The algorithms have been integrated into an interactive analog IC design environment customizable through an user friendly interface. Both tools are used in an SDL design flow to speed up analog ASIC design in modern mixed signal processes.

AR is a single layer maze router with a special oversizing algorithm and an advanced treatment of arbitrary polygons during connection phase. Due to its oversizing algorithm AR is able to generate diagonal path segments near obstacle corners which leads to a reduction of current density. GR is a two layer global router built upon AR which splits two layer routes into separate single layer parts each routable by AR. GR calculates all via positions and places the needed vias. The route done by GR and AR is customizable through an integrated reroute function after designer based via movements.

AR is currently used at R. Bosch GmbH to design analog and mixed signal ASICs and is commercially available to all users. The integration of GR into Bosch's SDL design flow is currently in progress.

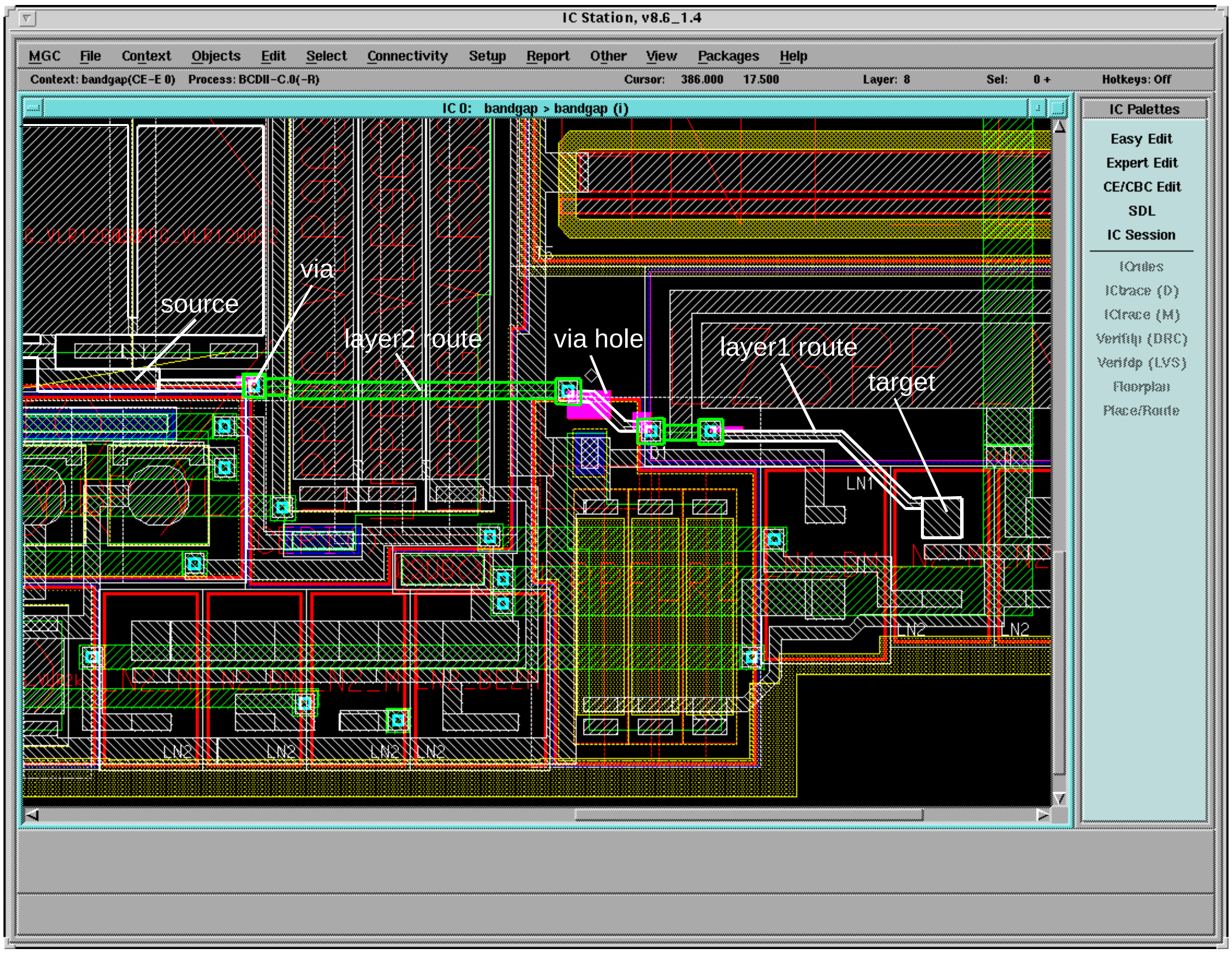


Figure 12: A routed example

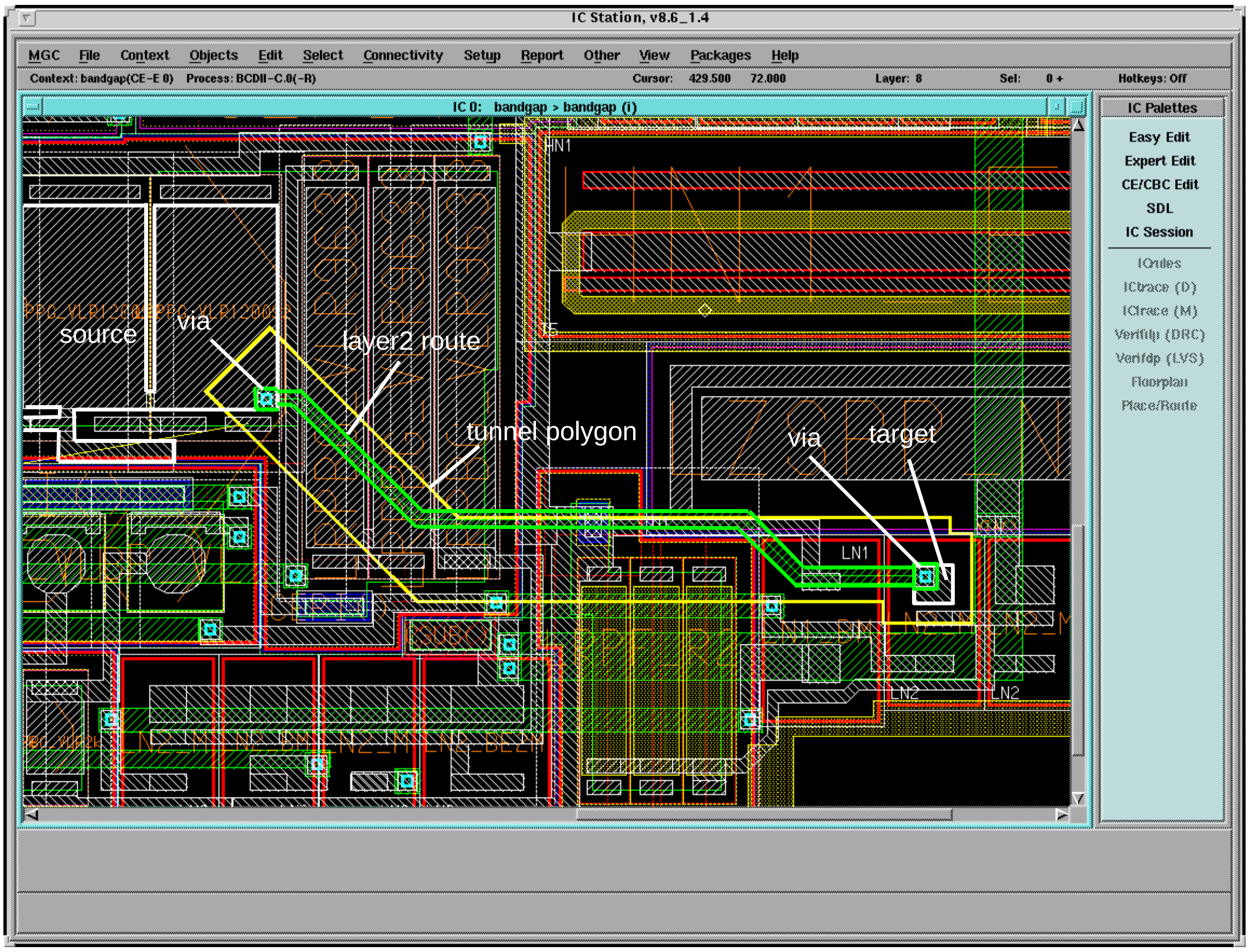

IC Station, v8.6_1.4
MGC File Context Objects Edit Select Connectivity Setup Report Other View Packages Help
Context: bandgap(CE-E 0) Process: BCDII-C.0(-R) Cursor: 429.500 72.000 Layer: 8 Sel: 0 + Hotkeys: Off
IC 0: bandgap > bandgap (i)
IC Palettes
Easy Edit
Expert Edit
CE/CBC Edit
SDL
IC Session
ICrules
ICtrace (D)
ICtrace (M)
Verifdp (DRC)
Verifdp (LVS)
Floorplan
Place/Route
source
via
layer2 route
tunnel polygon
via
target
LN1
LN2

Figure 13: Vias above source and target

References

[Car89] L.R. Carley et al, "ACACIA: The CMU Analog Design System", IEEE CICC, pp. 4.3.1-4.3.5, 1989

[DE0] DE 19530951.0, "Verfahren zur Anordnung von Leiterbahnen auf der Oberfläche von Halbleiter-bauelementen", C. Rödel, J. Scheible, applied for patent by R. Bosch GmbH, Stuttgart, 23.08.95

[DE7] DE 19531651.7, "Verfahren zur Anordnung von Leiterbahnen auf der Oberfläche von Halbleiter-bauelementen", C. Rödel, J. Scheible, applied for patent by R. Bosch GmbH, Stuttgart, 29.08.95

[DeG87] MGR DeGrauwe et al, "IDAC, an Interactive Design Tool for Analog CMOS Cicuits", IEEE JSSC, vol. SC-22, no. 6, pp. 1006-1116, Dec. 1987

[DoDu91] L.-O. Donzelle, P.-F. Dubois, "A new approach to layout of custom analog cells", Proc. EDAC, pp. 480-483, 1991

[Gie90] G. Gielen, K. Swings, W. Sansen, "An Intelligent Design System For Analogue Integrated Systems", Proc. EDAC, pp. 169-173, 1990

[Koh90] H. Koh, C. Sequin and P. Gray, "OPASYN: A Compiler for CMOS Operational Amplifiers", IEEE Trans. on CAD of IC, vol. 9, no. 2, pp. 113-125, Feb. 1990

[Lee61] C.Y. Lee, "An Algorithm for Path Connections and its Applications", IRE Trans. on Electronic Computers, pp. 346-365, 1961

[MzB93] V. Meyer zu Bexten et al, "ALSYN: Flexible Rule-Based Layout Synthesis for Analog IC's", IEEE JSSC, vol. SC-28, no. 3, pp. 261-268, 1993

[Oht86] T. Ohtsuki (editor), "Advances in CAD for VLSI: Layout Design and Verification", North-Holland, p. 110, ISBN 0 44 87894 7, 1986

[Rij89] J. Rijmenants et al, "ILAC: An Automated Layout Tool for Analog CMOS Circuits", IEEE JSSC, no. 2, pp. 417-425, April 1989

[Sch96] J. Scheible, "Device Generatoren und Autorouter für den Layoutentwurf analoger IC-Schaltungen in modernen Mischprozessen", 4. GMM/ITG-Diskus-sionssitzung, ANALOG '96, Oct. 1996, Berlin