AUTOMATIC ROUTING OF INTEGRATED CIRCUIT CONNECTIONS: A TUTORIAL

Peter M. Maurer
Department of Computer Science and Engineering
University of South Florida
Tampa, FL 33620

1. Introduction.

Before any two entities can communicate, there must exist a physical medium across which the communication can take place. In an integrated circuit, this medium takes the form of electrical connections between various parts of the circuit. Although these electrical connections can be created by hand, the process is time consuming, boring, and prone to error. For some circuits, creating the electrical connections (or *routing* as it is usually called) can take considerably more time than designing the active elements of the circuit. Because of this, the automatic routing was one of the first problems to be addressed by design automation. The area and timing (and hence the cost and performance) of an integrated circuit is directly dependent on the quality of the routing. Hence automatic routing has been extensively researched, and the body of literature is enormous. So much so that a list of just the basic references would require a bibliography much larger than this paper. In a paper of this size, it is impossible to give even a brief introduction all of the existing techniques. The reader is referred to [1] for a more detailed introduction to this subject, and a more extensive bibliography of the basic papers.

2. Definitions.

Before proceeding, it is necessary to define various terms that are in common use in the literature. This section will contain definitions of all terms used in the detailed version of this paper. The basic elements of the circuit are *blocks* (or *cells*) which are represented by rectangles at various locations in the XY plane. These are the objects to which connections will be made. Each block has one or more *terminals* which are represented by points, or sometimes line-segments, on the edge of the block. A terminal represents the location to which a connection must be routed. Terminals are labeled with the name of the *net* to which they belong. A *net* is a collection of terminals that must be wired together. Once the terminals have been connected the wiring is also considered to be part of the net. A net must have at least two terminals but there is no upper limit on the number of

terminals per net. Most routing algorithms assume that all terminals and wiring are aligned to a grid in the XY plane, and that all terminals and wires are of the same width. This allows the width of wires and terminals to be ignored. (However there is much research currently being done on "gridless" or "topological" routers.[2])

Physically a circuit consists of objects in several different *layers*, which provide a third dimension to the routing problem. Routing algorithms differ in the number of layers they assume to be present. Typically wire segments are assumed to lie in one of two layers of metal, while terminals may lie in three or more layers. Connections between layers are known as *vias* and are usually assumed to be the same width as terminals and wires. (In fact, vias must usually be *wider* than wire segments, and some routing algorithms seek to minimize the number of vias to produce a more compact routing.) Other definitions will be introduced as needed.

3. Types of Routing Algorithms.

Routing algorithms can be grouped into several broad categories, although there are some algorithms that are hard to classify. The first type of routing algorithms to be studied were called maze-routers, because they use maze-searching techniques to find paths between two terminals [3]. Figure 1 illustrates the search pattern for a typical maze router.

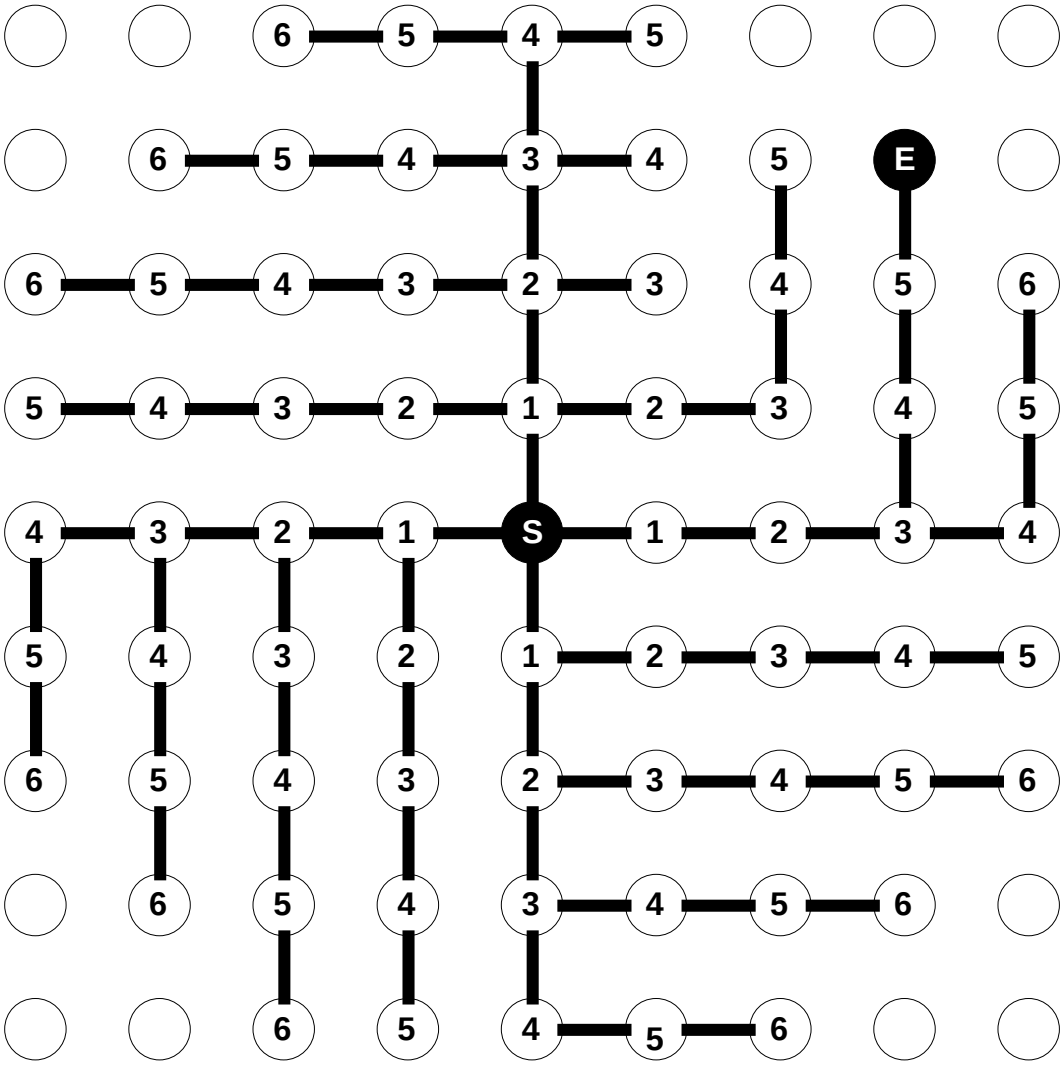

Figure 1. The Search Pattern for a Maze Router.

As can be seen from Figure 1, a maze router first searches the cells that are closest to the starting terminal, and then proceeds in a breadth first manner by searching the cells adjacent to those that have already been searched. When the connecting terminals have been found, the search stops and the wire is created. The precise path depends on the order in which neighboring cells are searched. Maze routers will always find a path, if one exists, and the path found is guaranteed to be of minimum length. However, these two points apply only to an individual net *at the time it is being routed*. Maze routers do not guarantee that the total length of all wires will be minimized, or that a complete routing for the entire circuit will be found, even if one exists. The problem is that nets routed early tend to block cells that are needed by nets routed later. This problem has been attacked in several different ways. Some algorithms will move blocking wires aside[4], or remove them entirely and attempt to reroute them later[5]. These last algorithms are usually called "rip-up" routers. Other algorithms allow the most difficult nets to be placed manually[6]. However, the most widely used approach

is to avoid the use of maze-routers altogether, or to use them to route those few nets that cannot be routed by other methods.

The channel router[7] is an efficient alternative to the maze-router, but it is designed to work in a more restricted environment. While maze-routers place no restrictions on the size or shape of the wiring area or on the position of the terminals, channel routers assume that the wiring area (called a *channel*) is rectangular with fixed terminals at the top and bottom and no terminals at the right or left. In spite of this, channel routers have become the workhorse of the IC industry, so a detailed discussion of these types of routers is warranted.

A typical channel is illustrated in Figure 2. The vertical lines represent terminals, while the numbers represent nets. By convention, a net number of zero represents an unused terminal. The simplest channel routers use two layers, one for vertical wire segments, and another for horizontal segments. A via is required at every right-angle bend. Since rotations in increments of 90 degrees are trivial, there are four physical orientations for each channel.

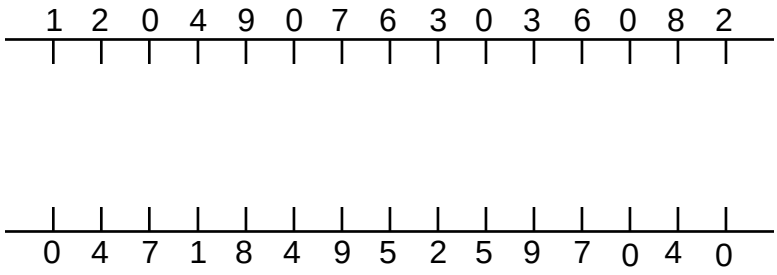

Figure 2. A Typical Routing Channel.

A channel is assumed to be laid out on a horizontal grid. There are terminals at each grid point on the top and bottom edges, but not all of these terminals are used. A net may have at most one horizontal segment (called a *trunk*) which extends from the rightmost terminal to the leftmost terminal. The horizontal row in which a trunk is placed is called a *track*. Depending on the application, the number of tracks may be fixed or variable. A trunk is connected to its terminals by vertical segments called *branches*. (By convention, this terminology is the same as that used in [8].) Figure 3 illustrates the routing of some typical nets. Note that since the trunks and the branches lie in different layers, there is no electrical connection between the leftmost branch of net 9, and the trunk of net 7.

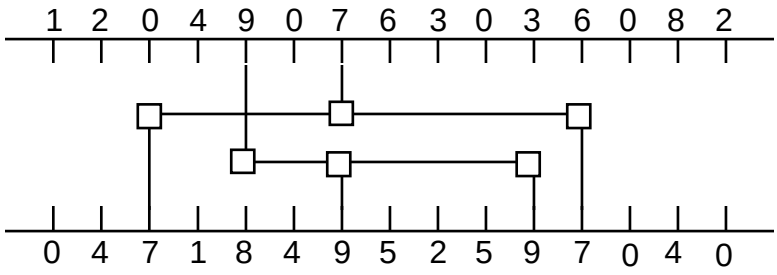

Figure 3. Some Typical Nets.

Note that in Figure 3, the trunk for net 7 must lie above the trunk for net 9, otherwise the two nets will be shorted when the center branches are attached. This is an example of a *vertical constraint*. Any column that contains terminals for two different nets produces one vertical constraint. The set of vertical constraints is usually represented as a directed graph with the nets represented by vertices and vertical constraints represented by directed arcs. Figure 4 presents the vertical constraint graph for the channel pictured in Figure 3.

Note that the vertical constraint graph of Figure 4 is acyclic. It is possible for some channels to have cyclic vertical constraint graph, in which case special techniques must be used to break the cycles.

The vertical constraint graph is used to implement the simplest of the channel routing algorithms, called the "left edge" algorithm[7]. This algorithm proceeds as follows.

1. Set the current track to 1. (Tracks are numbered from 1 starting with the topmost track).
2. Inspect the vertical constraint graph and obtain the set of all vertices that have no predecessors. (This is the set of nets that are routable in the current track.)
3. From the set of nets that are routable in the current track, find the net with the leftmost terminal, assign its trunk to the current track, and remove it from the set of nets routable in the current track.
4. Remove any nets whose trunks overlap with the most recently routed net from the set of nets routable in the current track. If any nets remain in this set, go back to step 3.
5. Delete all nets routed in the current track from the vertical constraint graph. If the graph is not empty, increment the current track number by 1 and go back to step 2.
6. Attach the branches and the vias, and map the logical routing into physical structures.

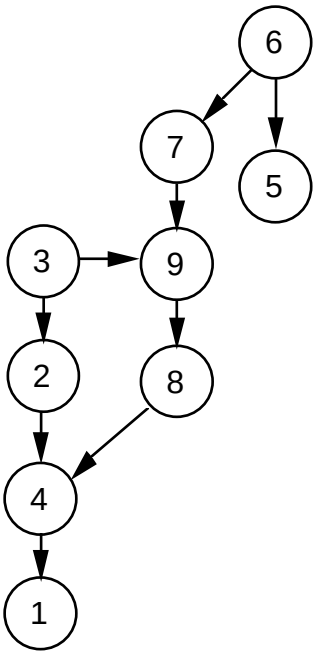

Figure 4. A Vertical Constraint Graph.

Note that the left-edge algorithm assumes that there are an unlimited number of tracks in the channel. It is, of course, trivial to add a check to see if the number of tracks in a fixed-width channel has been exceeded, but it is often more efficient to obtain a lower bound on the number of tracks required before routing begins. To do this, one draws a vertical line between any two terminals, and counts the nets that have terminals on both sides of the line. This count is called the *density* at that point. The *maximum density* or *channel density* is the maximum density over all vertical lines that could be drawn through the channel. The channel density is a lower bound on the number of tracks required to route the channel, but not necessarily the *absolute* lower bound. Unfortunately, the left edge algorithm is not guaranteed to use the minimum number of tracks. In fact. the problem of routing a channel in the minimum number of tracks is known to be NP-complete[9].

The channel routing problem must be solved heuristically, and there are heuristics that appear to work much better in practice than the left-edge algorithm. The simplest of these is doglegging[10]. A *dogleg* is a vertical wire segment that connects two trunks. Doglegging allows nets to be assigned to two or more trunks, which tends to produce more compact routings, than the "pure" left-edge algorithm. Doglegs are typically used only at the net's terminal positions, because this produces only one extra via. A dogleg at any other position would produce *two* extra vias. Furthermore, the nets that tend to cause problems are usually those that have many terminals, thus breaking trunks at terminal positions tends to reduce the harmful effects of such nets. Figure 5 illustrates the two nets of Figure 3 routed with doglegs.

Other techniques of note are the greedy channel router [11], the hierarchical channel router[12], and YACR2[13].

Regardless of which technique is used to route the channel, it is a simple matter to extend the algorithm so that a net can be routed out of the left or right edge, at an arbitrary position. It is also possible to extend the algorithm to route nets from the left edge to the right edge, even though the net has no terminals in the channel. The importance of these two features will be explained in the next section. It is often necessary to add a second routing step to the channel router, primarily to resolve cyclic vertical constraints. If a channel router cannot complete some nets (because of cyclic constraints) the unrouted nets can be completed using a maze router. This step is usually referred to as *cleanup routing*.

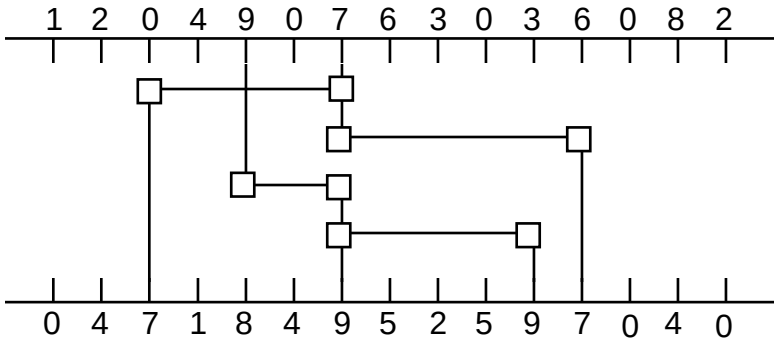

Figure 5. An Example of Dogleg Routing.

A variation on the channel router is the switch-box router. A *switch-box* is a rectangular area with *fixed* terminals on all four sides (as opposed to a channel which has floating terminals on the right and left). Switchbox routing tends to be more complex than channel routing, although several good algorithms have been developed[14,15].

There are many other classes of routing algorithms, most of which are designed to solve specialized problems such as power-and-ground routing or single layer routing[16]. A complete discussion of all the existing types of algorithms is beyond the scope of this paper.

4. Global and Detailed Routing

As was stated in the last section, channel-routing has become the workhorse of the IC industry. This is true in spite of the fact that the general IC routing problem deals with wiring areas that are vastly different from a simple channel. Figure 6 illustrates a simple IC floor plan. The rectangles represent blocks to be routed, while the space between the rectangles represents the wiring area.

Although the wiring area of Figure 6 is irregular, it can be partitioned into three channels as illustrated in Figure 7. After such a partitioning, a channel router can be used to route each of the channels. In figure 7 it is necessary to route channels A and B before routing channel C,because there may be nets with terminals in more than one channel. Suppose, for example, there is a net with

terminals in all three channels. The channel router is capable of extending the Channel-A and Channel-B trunks for these nets to the right and left edges of these channels, but the exact position where these trunks will touch the edge of the channel will not be known until channels A and B have been routed. Channel C cannot be routed until the location of these trunks is known. This can cause problems if channel definition is done in a naive way, as Figure 8 illustrates.

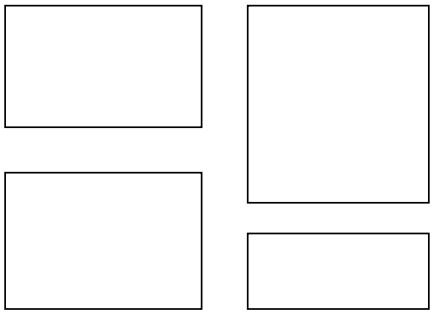

Figure 6. A Simple IC Floor plan.

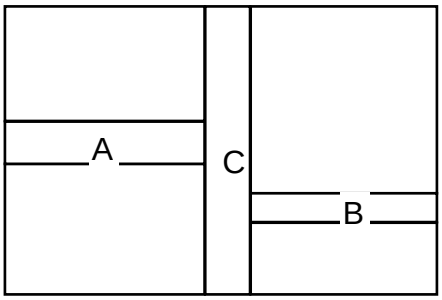

Figure 7. Partitioning an Irregular Area into Channels.

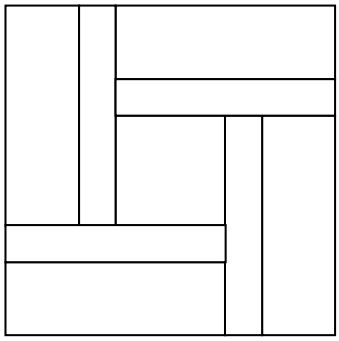

Figure 8. An Unroutable Channel Partitioning.

A number of channel-definition algorithms have been developed that deal effectively with floor plans such as that illustrated in Figure 8 [17].

After a reasonable set of channels has been defined, the next step is *global routing* of nets that have terminals in more than one channel[18]. It is often the

case that a net must traverse one or more channels in which it has no nets. When this occurs, there are usually several paths the net could take to reach the appropriate channels. Global routing is the process of assigning such nets to channels in such a way that all nets can be routed using a channel router. In addition to the obvious problem, global routers must also deal with channel congestion (assigning too many nets to a channel) and routing of critical nets. (A *critical net* is one whose timing constraints are severe, and must be routed along the shortest possible path.) Once global routing is complete, the channel router (or *detailed router* as it is called in this context) is used to complete the wiring of the IC. Although the channel router is the most popular choice for detailed routing, other types of routers can be used for this purpose.

Most global routing algorithms treat the global routing problem as a shortest-path algorithm. Some algorithms attempt to deal with the congestion problem dynamically, and will avoid channels that have become congested[19]. These algorithms tend to be sensitive to the order in which nets are routed. Other algorithms ignore the congestion problem and simply search for the shortest path [20]. In the second case it is usually necessary to use a second algorithm to reduce the congestion the original routing [21].

5. Improving Routing and Routability.

The routing techniques described in the previous sections treat blocks as "black-boxes" without internal structure. In reality, however, blocks have a complex internal structure that can be rearranged to improve the quality of the routing. An enormous amount of research has been done on the problem of placing components in such a way as to reduce the complexity of the wiring. There are three aspects to this problem, standard-cell placement, floor planning, and pin assignment. *Floor planning* is the process of placing rectangular blocks of differing sizes so as to minimize area and wiring complexity. This task has traditionally been done manually, and for the most part is still only partially automated[22]. (See, however, [23].) *Standard cell placement* is the process of automatically creating a circuit layout from a gate-level description of the circuit. *Pin assignment* is the process of selecting an appropriate location on a block for the terminals of a net.

Because pin assignment has been extensively used in the design of printed circuit boards, the terminology found in the literature is somewhat different than that used for routing algorithms. Nevertheless, pin assignment is an important step in reducing the wiring complexity of an integrated circuit. The two most important aspects of pin assignment are handling functionally equivalent terminals and taking advantage of equipotential terminals. Figure 9 illustrates the problem of handling functionally equivalent terminals.

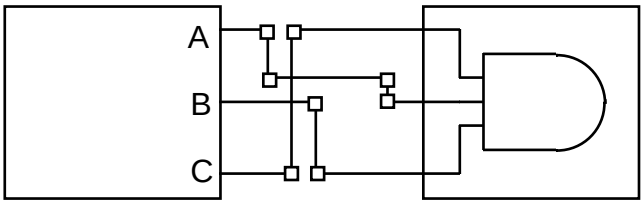

Figure 9. An Bad Pin Assignment.

In Figure 9, a bad pin assignment for the nets A, B, and C has resulted in an overly complex wiring of the nets. Since the inputs to the AND gate are functionally equivalent, these nets could be routed straight across. However with the pin assignment illustrated in Figure 9, the channel constraint graph is cyclic, requiring a dogleg in net A, and excessively long wires for all three nets.

The problem of equipotential terminals is illustrated in Figure 10

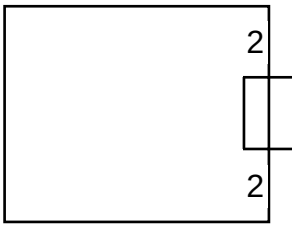

Figure 10. Equipotential Terminals.

In Figure 10, both terminals are assigned to net 2, but there is an internal connection inside the block between the terminals. Routing algorithms typically assume that identically labeled terminals must be connected by external wiring. However, when an internal connection between terminals exists, it is possible to leave one or more of the terminals unwired. More significantly, it is possible to take advantage of the internal wiring to complete nets that are difficult to wire externally.

Some of the better-known pin-assignment algorithms are the concentric circle algorithm[24], the nine-zone method[25], and the topological method[26].

Standard cell placement is probably the most extensively studied technique for reducing the wiring complexity of integrated circuits. Given a gate-level description of a circuit, the problem is to arrange the gates in such a way as to minimize the wiring complexity of the circuit. The circuits for the gates are usually assumed to be taken from a library of pre-designed circuits. Each such circuit is called a *standard cell*. Standard cells are assumed to have the form illustrated in Figure 11.

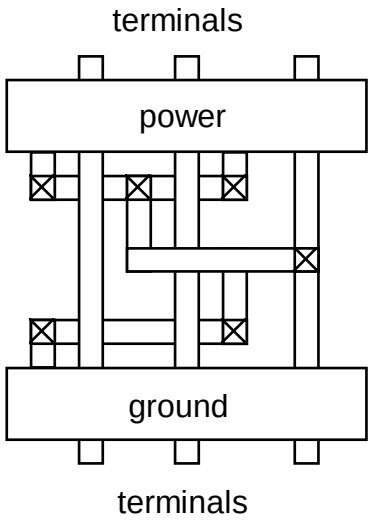

Figure 11. A Typical Standard Cell.

All standard cells are the same height, so that power and ground will abut properly when they are laid out in rows. The standard cells for a circuit are partitioned into several sets and laid out in rows to form a rectangular array. The space between the rows is used for channel routing. The complexity of the wiring is highly dependent on the partitioning and on the location of the cells within a row.

Many placement strategies have been studied. The most important of these are min-cut placement [27] and cluster growth[28]. Simulated annealing [29] and other techniques [30] have been used to improve the quality of initial placements obtained from one of the other techniques.

6. Conclusion

Although routing is one of the most thoroughly studied problems in computer-aided integrated circuit design, there are still many opportunities for future research. Because the problem must be attacked heuristically, there is a great deal of ongoing research to discover new and more effective heuristics. A potential area for new research is optimizing either the placement or wiring of an integrated circuit to enhance the performance of a collection of nets that represent a single communication channel. Placement and wiring improvement techniques have traditionally focused on total wire length as a figure of merit. Some techniques weight critical nets more heavily to guarantee that nets with severe timing constraints are routed by the shortest path. These techniques could possibly be extended to consider the characteristics of a collection of nets rather than treating each net individually. One approach that has gone in this direction is the routing technique used by Lincoln Laboratories to route wafer-scale circuits [31]. This technique collapses buses into single nets so that the individual nets that comprise the bus are routed along parallel paths. (Similar features are provided by many commercially available systems.)

Traditionally, routing has been implemented as a separate step rather than as an integrated part of the design process. (The same comment could be made about most phases of the design process.) This is not surprising when one considers the complexity of the routing problem. However, this trend is slowly changing, and it is likely that in the future many phases of the design process, including routing, will be more tightly integrated.

It is important to emphasize that this paper contains only the barest introduction to automatic integrated circuit routing. The reader is referred to the many references, especially [1] for more information. Current research on routing is reported in the IEEE Transactions on Computer Aided Design of Circuits and Systems, the annual ACM/IEEE Design Automation Conference, the International Conference on Computer Aided Design, and several other journals and conferences. The reader is referred to these sources for the most up to date information.

REFERENCES

1. Preas, B. and M. Lorenzetti (eds.), *Physical Design Automation of VLSI Systems*, The Benjamin/Cummings Publishing Co. Inc., Menlo Park, CA, 1988.

2. Lorenzetti, M., "Algorithms and Models for a Topologically Based Interconnection and Routing System," Ph.D. Dissertation, University of Texas at Austin, 1983.

3. Lee, C., "An Algorithm for Path Connections and its Applications," *IRE Transactions on Electronic Computers*, Vol. EC-10, pp. 346-365, Sept., 1961.

4. Scott, W, and J. Ousterhout, "Plowing: Interactive Stretching and Compaction in Magic," *21st Design Automation Conference*, pp. 166-172, 1984.

5. Dees, W., and P. Karger, "Automated Rip-Up and Reroute Techniques," *19th Design Automation Conference*, pp. 432-439, 1982.

6. Hightower, D., "A Manually Driven Line Search Router in a Line Editor,", *International Conference on Computer Aided Design*, pp. 276-278, 1985.

7. Hashimoto, A., and J. Stevens, "Wire Routing by Optimizing Channel Assignment within Large Apertures," *8th Design Automation Workshop*, pp. 155-163, 1971.

8. Sato, K., H. Shimoyama, T. Nagai, M. Ozaki, and T. Yahara, "A 'Grid-Free' Channel Router," *17th Design Automation Conference*, pp. 264-278, 1980.

9. Szymanski, T., "Dogleg Channel Routing is NP-Complete," *IEEE Transactions on Computer Aided Design*, Vol. CAD-4, pp. 31-41, Jan. 1985.

10. Deutsch, D., "A Dogleg Channel Router," *13th Design Automation Conference*, pp. 425-433, 1976.

11. Rivest, R., C. Fiduccia, "A 'Greedy' Channel Router," *19th Design Automation Conference*, pp. 418-424, 1982.

12. Burstein, M, and R. Pelavin, "Hierarchical Channel Router," *20th Design Automation Conference*, pp. 591-597, 1983.

13. Reed, J., Sangiovanni-Vincentelli, and M. Santomauro, "A New Symbolic Channel Router: YACR2," *IEEE Transactions on Computer Aided Design*, Vol. CAD-4, pp. 208-219, Jul. 1985.

14. Joobani, R. and D. Siewiorek, "WEAVER: A Knowledge-Based Routing Expert,", *IEEE Design and Test of Computers*, Vol.3, No. 1, pp. 12-23, Feb. 1986.

15. Cohoon, J., and P. Heck, "BEAVER: A Computational-Geometry-Based Tool for Switchbox Routing," *IEEE Transactions on Computer Aided Design*, Vol. 7, pp. 684-697, Jun. 1988.

16. Moulton, A., "Laying the Power and Ground Wires on a VLSI Chip," *20th Design Automation Conference*, pp. 754-755, 1983.

17. Kimura, S., N. Kubo, T. Chiba, and I. Nishioka, "An Automatic Routing Scheme for General Cell LSI," *IEEE Transactions on Computer Aided Design*, Vol CAD-2, pp. 285-292, Oct. 1983.

18. Clow, G., "A Global Routing Algorithm for General Cells," *21st Design Automation Conference*, pp. 45-51, 1984.

19. Li, J-T. and M. Marek-Sadowska, "Global Routing for Gate Array," *IEEE Transactions on Computer Aided Design*, Vol CAD-3, pp. 298-307, Oct. 1984.

20. Patel, A., N. Soong, R. Korn, "Hierarchical VLSI Routing - An Approximate Routing Procedure," *IEEE Transactions on Computer Aided Design*, Vol. CAD-4, pp. 121-126, Apr. 1985.

21. Vecchi, M., and S. Kirkpatrick, "Global Wiring by Simulated Annealing," *IEEE Transactions on Computer Aided Design*, Vol. CAD-2, pp. 215-222, Oct. 83.

22. Horng, C., and M. Lie, "An Automatic/Interactive Layout Planning System for Arbitrarily-Sized Rectangular Building Blocks," *18th Design Automation Conference*, pp. 293-300, 1981.

23. Otten, R. "Annealing Applied to Floor plan Design in a Layout Compiler," *Automation '86*, pp. 185-228, 1986.

24. Koren, N., "Pin Assignment in Automated Printed Circuit Board Design," *9th Design Automation Workshop*, pp. 72-79, 1972.

25. Mory-Rauch, L., "Pin Assignment on a Printed Circuit Board,", *15th Design Automation Conference*, pp. 70-73, 1978.

26. Brady, H. "An Approach to Topological Pin Assignment," *IEEE Transactions on Computer Aided Design*, Vol. CAD-3, pp. 250-255, Jul. 1984.

27. Breuer, M. "A Class of Min-Cut Placement Algorithms," *14th Design Automation Conference*, pp. 284-290, 1977.

28. Schuler, D. and E. Ulrich, "Clustering and Linear Placement," *9th Design Automation Workshop*, pp. 50-56, 1972.

29. Sechen, C. and A. Sangiovanni-Vincentelli, "The Timberwolf Placement and Routing Package," *Custom Integrated Circuits Conference*, May 1984.

30. Isopovici, A., C. King, and M. Breuer, "A Module Interchange Placement Machine," *20th Design Automation Conference*, pp. 171-174, 1983.

31. Frankel, R., J. Hunt, M. van Alstyne, and G. Young, "SLASH - An RVLSI CAD System," *International Conference on Wafer Scale Integration*, pp.31-38, 1989.