Technische Universität Berlin

FACHBEREICH 3
MATHEMATIK

SWITCHBOX ROUTING IN VLSI DESIGN: CLOSING THE COMPLEXITY GAP

by

STEPHAN HARTMANN MARKUS W. SCHÄFFTER
ANDREAS S. SCHULZ

No. 500/1996

Switchbox Routing in VLSI Design: Closing the Complexity Gap

Stephan Hartmann[*] Markus W. Schäffter[*] Andreas S. Schulz[*]

September 1996 / Update July 1997

Abstract

The design of integrated circuits has achieved a great deal of attention in the last decade. In the routing phase, there have survived two open layout problems which are important from both the theoretical and the practical point of view. Up to now, switchbox routing has been known to be solvable in polynomial time when there are only 2–terminal nets, and to be NP–complete in case there exist nets involving at least five terminals. Our main result is that this problem is NP–complete even if no net has more than three terminals. Hence, from the theoretical perspective, the switchbox routing problem is completely settled.

The NP–completeness proof is based on a reduction from a special kind of the satisfiability problem. It is also possible to adopt our construction to channel routing which shows that this problem is NP–complete, even if each net does not consist of more than five terminals. This improves upon a result of Sarrafzadeh who proved the NP–completeness in case of nets with no more than six terminals.

1 Introduction

Very large scale integrated circuit layout (VLSI) is one of the amazingly growing areas in discrete mathematics and computing science of the last years, due to both its practical relevance and its importance as a trove of combinatorial problems. Usually, in VLSI design one distinguishes between the phase of placing physical components and the subsequent routing phase realizing the conducting connections between them.

The routing phase itself consists of the layout problem and the corresponding layer assignment. We refer the reader to the book of Lengauer [Len90] and to the survey of Möhring, Wagner, and Wagner [MWW95] for a detailed description of this process as well as for comprehensive surveys of the use of combinatorial and graph–theoretical methods in VLSI design. Here, we concentrate on the layout problem where the course of the wires to connect the cells in a single plane has to be determined.

Most generally, the problem is to find an edge–disjoint packing of Steiner trees in a given planar graph. To be more precise, we are given a graph $G = (V, E)$, the so–called *routing graph*, and k sets $N_1, \dots, N_k \subseteq V$ called *nets*. In this context, the elements of the nets are referred to as *terminals*. The task is to find k pairwise edge–disjoint Steiner trees $T_1, \dots, T_k \subseteq E$ such that T_i connects the terminals of net N_i, if they exist, or to assert that there is no such packing. A solution of the Steiner tree packing problem is called *layout*. For planar graphs, Kramer and van Leeuwen [KvL84] showed that the Steiner tree packing problem is NP–complete even if there are only two–terminal nets. Korte, Prömel and Steger [KPS90] complemented their result by proving the NP–completeness of the problem if there are only two multi–terminal nets. If all terminals are assigned to the outer face of the routing graph, Okamura and Seymour [OS81] gave sufficient conditions for instances that can be solved in polynomial time.

The routing graphs arising in VLSI design are actually very special planar graphs. Most frequently, they are rectangular grids, corresponding to the usual shape of the physical layout areas. Such routing problems have been

[*]Technische Universität Berlin, Fachbereich Mathematik, MA 6–1, Straße des 17. Juni 136, 10623 Berlin, Germany, E-mail hartmann/shefta/schulz@math.TU-Berlin.DE

attacked by quite different methods ranging from purely bottom–up methods over floor–planning techniques up to polyhedral combinatorics (see, e.g., [Len90, GMW93]). There are two types of problems on a grid which are of particular importance, namely *switchbox routing* and *channel routing*. In both cases, all terminals are placed on the boundary of the grid. In switchbox routing the terminals may be placed on all four sides. Channel routing is a special case of switchbox routing where the terminals are only placed on the lower and the upper side of the grid.

For the switchbox problem, Preparata and Mehlhorn [MP86] gave a polynomial time algorithm that constructs a layout, if all nets contain only two terminals. For the channel routing problem, Sarrafzadeh [Sar87] proved the NP–completeness if some of the nets involved have six or more terminals. He also claimed (without giving a proof) the NP–completeness of problems involving nets with at least five terminals. This implies the same result for switchbox routing. We show that switchbox routing is NP–complete even if all nets have at most three terminals. Hence, the present paper closes the gap between the algorithm of Preparata and Mehlhorn on one side, and the NP–completeness result of Sarrafzadeh on the other side. As a consequence, heuristic algorithms are of interest for all instances of the switchbox routing problem that contain nets with more than 2 terminals. An overview of different heuristics can be found in [MS92]. It is also possible to transfer our construction to channel routing. This results in an NP–completeness proof for the case that every net has at most five terminals, see [Har96]. We would like to mention that our reduction is partially based on refinements of some of the ideas in [Szy85] and [Sar87].

The paper is organized as follows. In Section 2, we define the 3–bounded 3–SAT problem and the switchbox routing problem and give a first, introductory description of the transformation. The following sections discuss all the details of the transformation. In Section 5, we prove the correctness of our result. We conclude with some remarks in Section 6.

2 A First Description of the Reduction

An instance of the switchbox routing problem consists of a *routing region* and a set of *nets*. The routing region is assumed to be a rectangular grid, called *switchbox*, with n vertical lines and m horizontal lines, also called *tracks*. The set of nets consists of k nets $N_1,..,N_k$, where each net is a set of so–called *terminals* which here are intersection points at the boundary of the grid.

A solution of the switchbox routing problem, called a *layout* or a *routing*, is given by pairwise edge–disjoint Steiner trees $T_1,...,T_k$ embedded in the grid such that T_i connects the terminals of net N_i, $i = 1,\dots,k$. In the layout, all induced paths must have disjoint edges but they may meet at the intersection points of the grid. In VLSI design, this is called the *knock–knee model* since at an intersection point, two induced paths may cross or both may change their direction (forming a double–bend, called knock–knee). In contrast to the knock–knee model, there is the *Manhattan model* where only crossings but no knock–knees are allowed. For the Manhattan model, Szymanski [Szy85] showed that channel routing with 4–terminal nets is NP–complete and hence so is switchbox routing. This result is extended by Middendorf [Mid93] who showed that even the 2–terminal channel routing problem in the Manhattan model is NP–complete. In the following, we consider the knock–knee model.

The 3–terminal switchbox routing problem

Instance: A rectangular routing region consisting of n vertical and m horizontal lines. A collection $\{N_1,..,N_k\}$ of nets, each net consists of at most 3 terminals. The terminals are assigned to the intersection points at the boundary of the grid.

Question: Is there an edge-disjoint knock-knee routing for the nets in the given routing region?

Before explaining the basics of our reduction, we introduce some notions which prove useful in the discussions to follow. We number the horizontal lines of the grid top–down and the vertical lines from the left to the right. The segment of the grid between the vertical lines with indices i and $i+1$ is called a *column* and is denoted by $\vec{\imath}$. The *(local) horizontal density* $d_h(i)$ of column $\vec{\imath}$ is defined as the number of nets that have to cross column $\vec{\imath}$, i.e., it is the number of nets that contain terminals at the left–hand side as well as terminals at the right–hand side of the column $\vec{\imath}$. We call the vertical line with index i *density–increasing* if $d_h(i) > d_h(i-1)$ and *density–decreasing* if $d_h(i) < d_h(i-1)$. Otherwise, we call the vertical line *density–preserving*. We call the number m of horizontal lines the *horizontal capacity* and the number n of vertical lines the *vertical capacity* of the switchbox. The *free horizontal capacity* of a column is the difference of the capacity minus the density of the column.

We interchangeably use the term net for a set of terminals and for the realization of its Steiner tree in the layout. The respective meaning should always be clear from the context.

The following is our main theorem.

Theorem 2.1. *The 3-terminal switchbox routing problem is NP–complete.*

The rest of the paper is devoted to prove this result. It is clear that 3–terminal switchbox routing is in NP. The reduction is from a special case of the 3–SAT problem.

The 3–bounded 3–SAT problem

Instance: A set $\mathcal{X} = \{x_1, \dots, x_N\}$ of Boolean variables, and a collection $\mathcal{C} = \{C_1, \dots, C_M\}$ of clauses over $\mathcal{X}$. Thereby, for each variable x_i there are at most three clauses in $\mathcal{C}$ that contain either x_i or $\overline{x}_i$. Moreover, each clause contains at most three literals.

Question: Is there a Boolean assignment to the variables in $\mathcal{X}$ such that every clause in $\mathcal{C}$ is satisfied?

The 3–bounded 3–SAT problem is NP–complete [GJ79, p. 259]. Without loss of generality, we assume that all clauses contain strictly more than one literal and that every Boolean variable occurs negated as well as unnegated.

The main ideas of the transformation are as follows.

- For each Boolean variable $x_i \in \mathcal{X}$ we introduce two nets X_i and $\overline{X}_i$, called *real–nets*. If x_i occurs in two clauses, X_i and $\overline{X}_i$ consist of two terminals each, otherwise they have three terminals.
- A variable x_i is meant to be TRUE if and only if net X_i is routed below net $\overline{X}_i$ in the layout, and FALSE otherwise.
- In order to maintain the relative vertical ordering of the real–nets X_i and $\overline{X}_i$, we aim to keep the horizontal density high throughout the switchbox. Therefore, we introduce so–called *extension–nets*. Each of these nets has exactly one terminal on the left or the right boundary of the grid. Their functionality is discussed in Section 3.
- We also force the real– and the extension–nets to certain tracks. This is the task of *sandwich–nets*. The details are given in Subsection 4.4.
- Each clause C_j is modeled as a block B_j of consecutive vertical lines. The constructed switchbox consists of these clause blocks chained from the left to the right (see Figure 1).

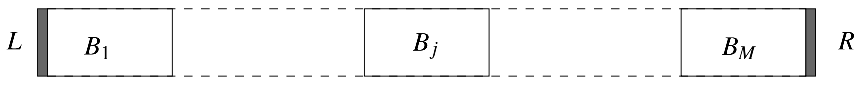


Figure 1: The coarse–structure of the resulting switchbox instance.

- The link of the Boolean variables to the clauses in which they occur is essentially captured as follows. If the Boolean variable x_i appears in the clause C_j, both nets X_i and $\overline{X}_i$ have each exactly one terminal on the same vertical line (called the *variable–line*) of the clause block B_j.
- This variable–line is surrounded by a certain collection of vertical lines (and nets placed on these lines) to guarantee that the terminals of X_i and $\overline{X}_i$ can be connected to its corresponding nets, respectively. This ensures that each literal can be TRUE or FALSE. The precise structure of clause blocks is described in Section 4.
- Throughout the paper, repeatedly occurring collections of vertical lines and nets of the same topological structure are called modules. The collection used in the previous item (which provides the free capacity needed) is called a *detour module* and is explained in Subsection 4.2.
- We use *gate modules* (see Subsection 4.3) in order to ensure that more than the available capacity is needed, in the case that all literals of a clause are FALSE.

The resulting instance of the switchbox routing problem is quite complex. To be precise, a 3–bounded 3–SAT instance consisting of N Boolean variables and M clauses is transformed into a switchbox routing problem instance with $n = 8N + 16M_2 + 31M_3$ vertical lines, $m = 10N + 10M_2 + 20M_3 + 4$ horizontal lines, and $k = 14N + 16M_2 + 32M_3 + 4$ nets, where M_2 denotes the number of 2–literal and M_3 the number of 3–literal clauses of the 3–bounded 3–SAT instance, respectively.

3 The Sides of the Switchbox

In this section, we introduce all tracks of the switchbox instance, explain to which nets they are dedicated to, and finally assign terminals to their left and right endpoints. Roughly, we can distinguish between three different areas of tracks. The top area is dedicated to the real– and extension–nets, the middle one to the gate–nets (see Subsection 4.3), and the bottom one to the detour–nets (see Subsection 4.2).

We insert two horizontal lines for each Boolean variable x_i, called *variable–tracks*. They are designated for the associated real–nets. Since the terminals of the real–nets only appear within clause blocks, we need to keep the variable–tracks occupied between the left–hand side (right–hand side, respectively) of the switchbox and the left–most (right–most) terminals of the real–nets. This is the task of the extension–nets. There is one extension–net for each real–net associated with a Boolean variable and for each side of the switchbox. Hence, there are four of them for each Boolean variable x_i which are denoted by $X_{i,lt}$, $X_{i,lb}$, $X_{i,rt}$, and $X_{i,rb}$, respectively. Here, l and r stand for left and right, and t and b stand for top and bottom, respectively.

Each extension–net has exactly three terminals. One is directly placed on the left (right) boundary of the grid and two of them are in the clause block which contains the left–most (right–most) terminal of the associated real–net. The terminals in a clause block are combined with a so–called clamp module which is discussed in detail in Subsection 4.4.

To explain the precise dedication of the introduced tracks and the arrangement of the terminals on the sides, we fix an arbitrary ordering of the clauses and re–number the Boolean variables correspondingly. The variables are numbered from 1 to N in order of their first occurrence with respect to the chosen clause ordering (from left to right).

Starting from the top, we assign the terminals of the extension–nets in the order $1, \dots, N$ to the sides of the switchbox. Thereby, the associated terminals of the upper and the lower extension–nets alternate. Each of these terminals is surrounded by a certain number of sandwich–nets. Sandwich–nets are also assigned to gate– and detour–nets. They have a special structure: they consist of three terminals; two of them are placed at the left–hand and at the right–hand side of the switchbox, respectively, both on the same track. Because of lack of capacity, we will see that sandwich–nets are forced to occupy the whole track to which their outer terminals are assigned to.

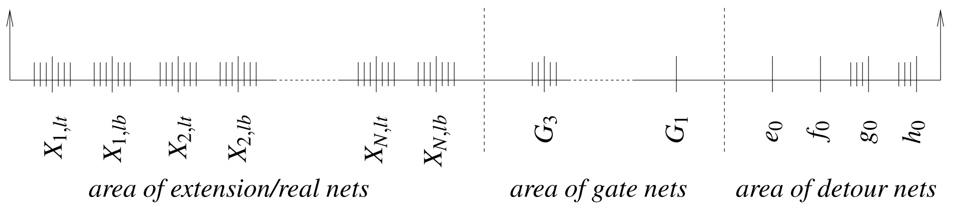


Figure 2: The left side L of the switchbox (in landscape mode).

Figure 2 depicts the terminal assignment at the left side of the switchbox. The terminals of extension–, gate– and detour–nets are represented by long dashes while the terminals of sandwich–nets, assigned to these nets, are represented by short dashes. The terminal assignment of the right side is similar. Note that all intersection points at the left and the right boundary of the grid are occupied by terminals.

4 The Clause Block

In this section, we explain the precise structure of a clause block. Every clause block consists of vertical variable–lines for every Boolean variable of the clause. The variable–lines are surrounded by detour modules which are themselves embedded into gate modules. We present these concepts as well as the clamp modules which serve to maintain the vertical ordering of the nets in the following subsections.

4.1 The Variable–Line

For a given clause C, we introduce a variable–line for each variable occurring in C. If a variable x appears unnegated in C, we place a terminal of net X at the lower position of the variable–line of x and a terminal of net $\overline{X}$ at the upper position. If x appears negated in C, the assignment is the other way around (see Figure 3).

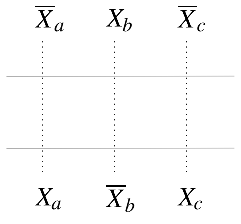


Figure 3: Variable–lines for $C = \{x_a, \overline{x}_b, x_c\}$

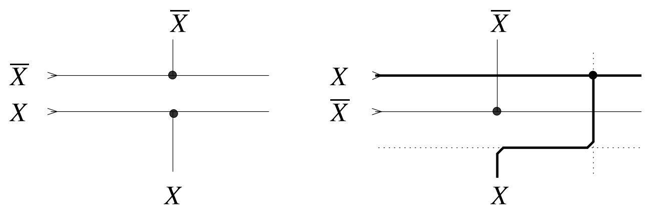


Figure 4: a) TRUE–routing b) FALSE–routing.

So far, we have mainly been concerned with the construction of the switchbox instance. To improve the accessibility of the discussions to follow, we turn for a moment to the interpretation point of view. As mentioned before, a variable x is meant to be TRUE if and only if net X is routed below net $\overline{X}$. Now, this is rendered more precisely. Let I denote the interval between the left–most and the right–most terminals of the real–nets X and $\overline{X}$. Let I^* be the maximal sub–interval of I such that the left– and the right–most vertical line is at maximum density. The variable x is defined to be TRUE if and only if net X is routed below net $\overline{X}$ in I^*, and FALSE otherwise. (The reason for the restriction of the definition to I^* is given in Section 5.)

Given the terminal assignment at the variable–lines and the interpretation of the routing of the real–nets, we can distinguish two possible routing types at each variable–line. If a variable x is TRUE (FALSE, respectively) and appears negated (unnegated) in the clause under consideration, then not both terminals of the associated variable–line can directly be connected to their dedicated track by use of the capacity of the variable–line only. Consequently, additional horizontal and vertical capacity is needed to route the necessary detour (see Figure 4 b). Such a routing is called a FALSE–routing. Every other kind of local routing (see, for instance, Figure 4 a) is called a TRUE–routing. It corresponds to a literal with value TRUE.

In order to allow a TRUE– or a FALSE–routing at every variable–line, the detour modules provide the required horizontal and vertical capacity.

4.2 The Detour Module

Detour modules appear around each variable–line. They are intended to provide the capacity needed for a FALSE–routing and, at the same time, to keep the horizontal and vertical density high enough to prevent a change of the vertical ordering of the nets. We distinguish between two types of detour modules, namely *right–handed* and *left–handed* ones. Since they are symmetric about the associated variable–line, we restrict ourselves to the description of right–handed detour modules.

Consider, say, the ith detour module. This module consists of four vertical lines, of two terminals of sandwich–nets, and of six terminals of four so–called *detour–nets* e_{i-1}, e_i, g_{i-1}, and g_i. These detour–nets serve to keep the capacity low which is caused by the special terminal assignment at the introduced vertical lines. The first terminal of the net e_{i-1} and the first two terminals of the net g_{i-1} are located in the previous detour module. The third terminal of g_{i-1} is placed on the upper endpoint of the first vertical line of the ith detour module. We put this line itself to the left of the variable–line surrounded by the ith detour module. The other three vertical lines of this detour module are placed to its right (see Figure 5). Terminals of the detour–net g_i are assigned to the bottom endpoints of the second and the fourth of these vertical lines whereas the bottom endpoints of the first and the third vertical line are used for the nets e_{i-1} and e_i, respectively. The remaining upper endpoints of the vertical lines to the right of the variable–line are designated for terminals of the nets S_x, e_{i-1}, and S_g, respectively. Here, S_x denotes a sandwich–net whose remaining two terminals are placed directly above the associated extension–nets of the Boolean variable x at both sides of the switchbox instance (see Figure 2). The other two terminals of the sandwich–net S_g are directly placed above the associated detour–net at both sides of the switchbox instance (see again Figure 2). The exact terminal assignment within the detour module can be taken from Figure 5.

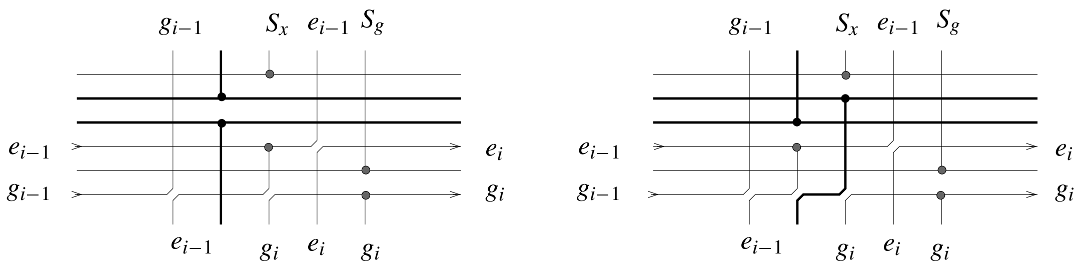


Figure 5: A layout for the detour module: a TRUE– and a FALSE–routing (left/right).

The vertical line immediately to the right of the variable–line is called the *detour–line*. The described detour module is called right–handed since the detour–line is on the right side of the variable–line. A left–handed detour module is obtained by reversing the terminal assignment from left–to–right. The detour–nets of a left–handed module are denoted by f_i and h_i instead of e_i and g_i, respectively.

The name detour–line is caused by its functionality. The detour caused by a FALSE–routing cannot be realized without using the vertical capacity of this line. For the same reason, the detour module provides free horizontal capacity of one track between its first vertical line, which is density–decreasing, and its detour–line which is density–increasing. The remaining vertical lines of the detour module, however, are density–preserving. The capacity provided by the detour module can be used to realize a TRUE– as well as a FALSE–routing. Notice that the free horizontal capacity of one track must be used in both cases (see Figure 5).

Finally, we should mention that, due to their exposed location, the detour–nets e_0, f_0, g_0, h_0 as well as $e_{M_2+2M_3}$, $f_{M_2+2M_3}$, $g_{M_2+2M_3}$, and $h_{M_2+2M_3}$ need a special treatment. First, the terminals of e_0, f_0, g_0, and h_0 which have not been assigned so far are placed on the left side of the switchbox, as already indicated in Figure 2. Similarly, the remaining terminals of the detour–nets $e_{M_2+2M_3}$, $f_{M_2+2M_3}$, $g_{M_2+2M_3}$, and $h_{M_2+2M_3}$ are assigned to the right side of the switchbox. Second, in contrast to the other detour–nets, the detour–nets g_0, h_0, $e_{M_2+2M_3}$, and $f_{M_2+2M_3}$ have two terminals only.

4.3 The Gate Module

In order to ensure that each clause will be satisfied by the variable assignment deduced from a layout, we have to guarantee that not at every variable–line within a clause a FALSE–routing can be realized. For this purpose, we introduce gate modules.

Each gate module consists of six vertical lines, four sandwich–nets and four *gate–nets* which consist of three terminals each. To simplify notation, we focus on one gate module and denote its corresponding gate–nets by G_i, $i = 1,..,4$. One terminal of each of these nets is assigned to the sides of the switchbox, the respective terminals of G_1 and G_3 to the left side, the ones of G_2 and G_4 to the right side (see Figure 2).

A gate module is based on the structure of so–called *autonomous intervals* which are introduced in [FWW93]. Autonomous intervals enforce detours, that is, the associated nets cannot be routed without using additional vertical and horizontal capacity. Figure 6 depicts possible routings of an autonomous interval. Such an interval is assigned to the third and the fourth vertical line of each gate module. The first and the last vertical line of the gate module are density–preserving. The second vertical line is density–decreasing whereas the fifth vertical line is density–increasing. Consequently, the free horizontal capacity is one between these vertical lines.

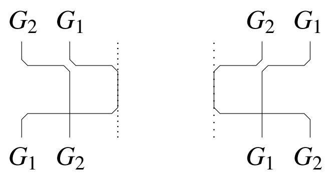


Figure 6: Layouts for an auton. interval.

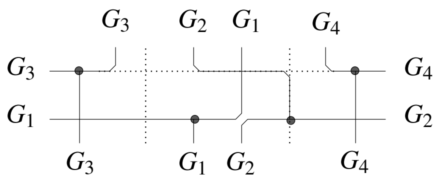
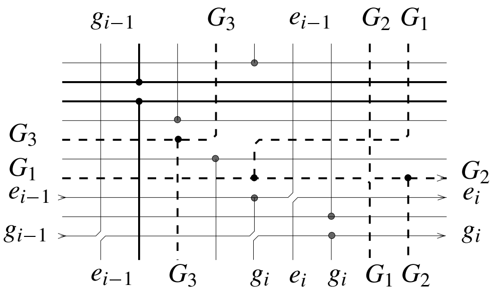


Figure 7: Structure of the gate module.

The precise terminal assignment of the involved gate– and sandwich–nets to these vertical lines is given in Figure 10, after the introduction of the clamp modules. Figure 7 depicts the pure structure of a gate module.

Two detour modules are embedded into one gate module. Their detour–lines are placed between the second and the third vertical line of the gate–module and between the fourth and fifth vertical line, respectively. In Figure 7, the embedded detour–lines are represented by dotted lines. For 2–literal clauses, the combination of gate– and detour–modules is as follows: we assign a right–handed detour module to the left variable–line and a left–handed detour module to the right variable–line; a gate module lies in–between. For 3–literal clauses, we first assign two detour modules to the variable–line in the middle: a left–handed one and a right–handed one. The left variable–line of the corresponding clause block is embedded into a right–handed detour module whereas the right one is embedded into a left–handed detour module. Afterwards, two gate modules are placed between the first two and the last two detour–lines, respectively. The interaction between the detour and the gate modules is illustrated in Figure 8.

In order to route the detour enforced by a gate module, the free horizontal capacity provided by the gate module itself and the capacity of at least one embedded detour–line must be used. This guarantees that not at every variable–line a FALSE–routing can be realized. In our interpretation, this means that at least one literal of the corresponding clause has to be TRUE. This will be proved in Lemma 5.6.



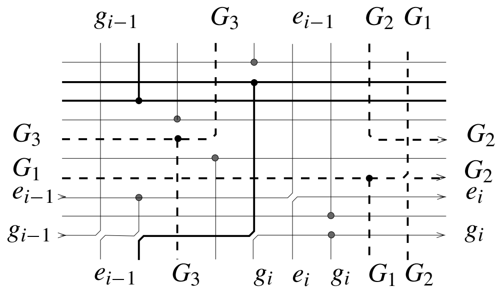
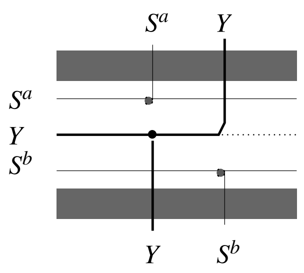


Figure 8: The interplay between gate and detour modules: a TRUE– and a FALSE–routing.

4.4 The Clamp Module

The crux of the transformation is to guarantee a certain vertical ordering of the nets. Therefore, we force nets to dedicated tracks. This is the task of *clamp modules*. A clamp module consists of two consecutive vertical lines, of two middle terminals of sandwich–nets, and of two terminals of a net which either starts or ends within a clause block and which should be forced to a dedicated track. Clamp modules are assigned to the gate–nets G_3 and G_4, and to each of the extension–nets.

Consider, say, a net Y whose left–most terminal is placed on the left side of the switchbox. The terminals of the associated sandwich–nets S^a and S^b (where a stands for above and b for below) are directly placed above and below the left–most terminal of Y at the left side and are assigned to the same positions at the right side of the switchbox. The first vertical line of the clamp module is density–preserving and the second is density–decreasing. The terminals of S^a and Y are assigned to the top, and the terminals of Y and S^b are placed on the bottom side of these lines, in this order. If a clamp module is assigned to a starting net whose right–most terminal is assigned to the right side of the switchbox, the first vertical line of the clamp module is density–increasing and the second is density–preserving. For the exact terminal assignment of a clamp module, we refer the reader to Figure 9. We will show in Proposition 5.2 that an ending net (starting net, respectively) is forced to its dedicated track because of the interaction of its terminal at the left (right) side of the switchbox, of the clamp module at its right (left) side, and the lack of capacity in–between.



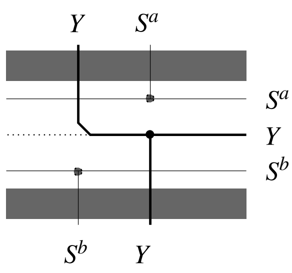


Figure 9: A clamp module assigned to an ending/starting net (left/right).

To conclude with the description of the construction, we give the positions of the clamp modules inside the clause blocks. The clamp module assigned to the gate–net G_3 (G_4, respectively) is placed at the left (right) side of the left (right) detour–line of the corresponding gate module (see Figure 7).

For the clamp module of an extension–net, we distinguish four cases according to the side (left or right) the third terminal of the extension–net is assigned to and to the type (right– or left–handed) of the detour module that surrounds the variable–line at the transition from the extension–net to the associated real–net. For reasons of symmetry, we may restrict ourselves to a right–handed detour module. If the variable–line contains the left–most terminals of the real–nets X_i and $\overline{X}_i$, then the clamp modules assigned to the extension–nets $X_{i,lt}$ and $X_{i,lb}$ are both placed to the left of this variable–line (see Figure 10). If the variable–line contains the right–most terminals of the real–nets, then the clamp module of the lower extension–net $X_{i,rb}$ is placed right to the variable–line. The clamp module of $X_{i,lt}$ is placed to the right of the associated detour–line.

At this point, all ingredients of the construction are described in detail.

4.5 An Example for a Clause Block

In this subsection, we give an example illustrating the quite complex structure of a clause block. The main principle is that most of the nets have the same routing in every layout, if one exists. This "skeleton" consists of all sandwich–, detour–, and extension–nets. Within this skeleton, the routings of the real– and gate–nets have to be realized.

Figure 10 essentially depicts the complete clause block for the 2–literal clause $\{\bar{x}_1, x_2\}$. We give all vertical lines, but tracks associated with nets from other clause blocks are only indicated by through shaded lines. It contains the routing of all nets of the "skeleton", i.e., nets that have the same routing in every layout. Note that only one pair of extension–nets occurs since we assume that only the real–nets of variable x_2 start in this clause block. The remaining nets marked with an arrow may have different routings (in the shaded rectangles) depending on different variable assignments.

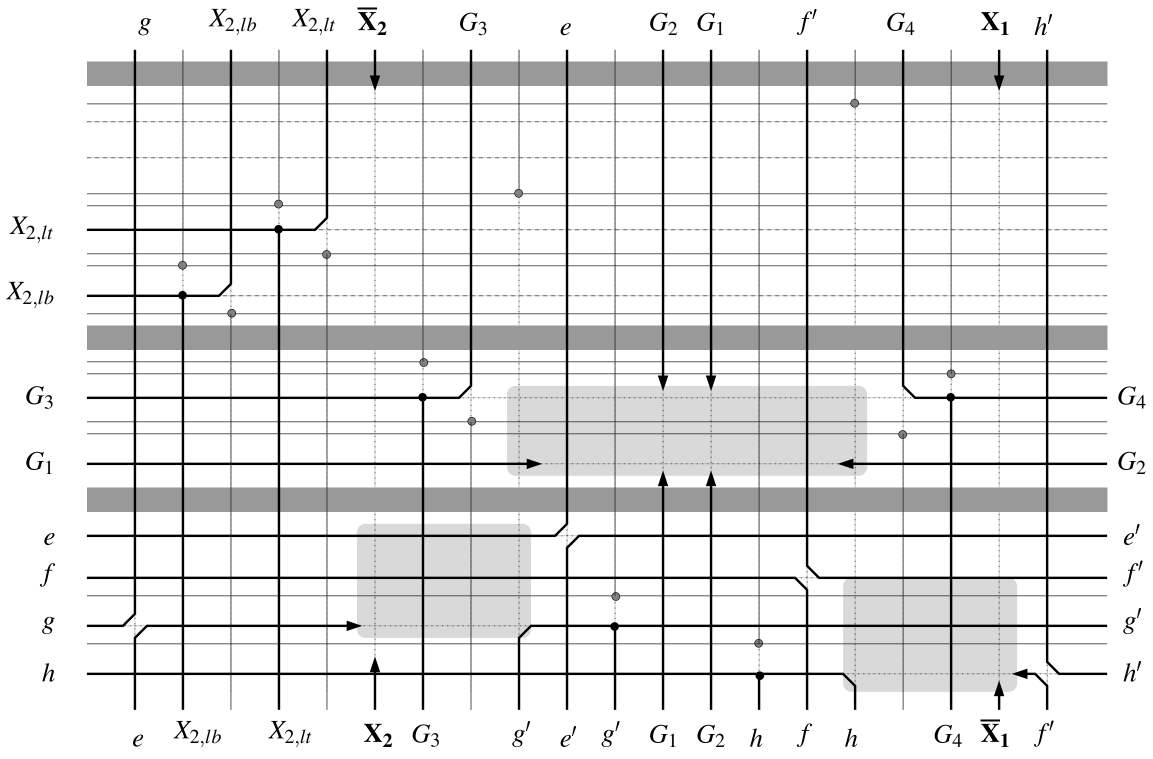


Figure 10: An example clause block for $C = \{\bar{x}_1, x_2\}$.

5 The Proof

At this point, the reduction from an instance of the 3–bounded 3–SAT problem to an instance of the 3–terminal switchbox routing problem is completely described. The NP–completeness proof has the following structure: Lemma 5.1 shows that a satisfying variable assignment for an instance of the 3–bounded 3–SAT problem induces a layout for the resulting switchbox routing problem. For the reverse direction, it is most important to show that the determination of the values of the Boolean variables from the layout is well defined. This is captured by Lemma 5.5. Once this lemma is proved, it remains to show that the assignment deduced from the layout satisfies all clauses of the underlying instance of the 3–bounded 3–SAT problem. This is done in Lemma 5.6.

Lemma 5.1 (Existence of a Layout). *Every feasible solution to an instance of the 3–bounded 3–SAT problem induces a layout for the resulting instance of the switchbox routing problem.*

Proof. Given an instance of the 3–bounded 3–SAT problem and a satisfying variable assignment, we now describe how to obtain a layout for the resulting switchbox routing problem instance.

1. Route each sandwich–net along the track where its left and right terminals are assigned to and connect its middle terminal using the vertical line it is assigned to.
2. Route all extension– and upper gate–nets along their dedicated tracks.
3. Assign the first two real nets X_1 and $\overline{X}_1$ to the highest free tracks. If variable x_1 is TRUE, route $\overline{X}_1$ above X_1, otherwise route them in reverse order. Continue in the same manner with the remaining real–nets.
4. Connect the terminals of the real–nets whenever the assignment of the terminals in the clause blocks corresponds to the ordering of the track assignment at the corresponding variable–lines. These are exactly the TRUE–routings.
5. Now route the lower gate–nets and their detours enforced by the autonomous intervals. Route these detours to the side where the routing of the variable–line was already done in Step 4. This is possible as at least one TRUE–routing per clause block exists (see Figure 8). For a 3-literal clause, this means if only the left–most (right–most) variable–line is routed, route the detours of both gate modules to the left–hand (right–hand) side. If only the variable–line in the middle is already routed, then route the detour of the left gate module to the right–hand side and vice versa.
6. Connect the remaining terminals of the real–nets on the upper side directly to their corresponding tracks. Connect the remaining terminals of the real–nets on the lower side by using the horizontal capacity in the area of the detour–nets and the vertical capacity of the detour–line as depicted in Figure 5, right side. These FALSE–routings can be realized since in each clause block the number of the remaining detour–lines is greater than or equal to the number of FALSE–routings.
7. Route the remaining detour–nets in a canonical way (see Figure 5).

This procedure yields a layout for the resulting switchbox routing problem. □

Proposition 5.2 (Clamp Module). *For every layout of an instance of the switchbox routing problem, all starting (ending) nets that are combined with a clamp module are routed along their dedicated tracks.*

Proof. Due to the symmetry of a clamp module, we consider without loss of generality a starting net Y. This means that the first vertical line of the clamp module is density–increasing and that the density is maximum at the next two columns (see Figure 9, right side). Let i denote the track dedicated to net Y and j denote the track net Y is actually assigned to. The tracks $i-1$ and $i+1$ are completely occupied by the sandwich–nets S^a and S^b which are assigned to the clamp module of net Y.

First, we assume that net Y is routed to a track $j < i-1$. (Recall that the tracks are numbered from the top to the bottom.) Then, the terminal assignment at the second vertical line of the clamp module forces either net Y or net S^a to occupy two tracks. But because of the maximum density at this line, each net must only occupy exactly one track. Hence, net Y cannot be routed along a track $j < i-1$. The similar reasoning applies if net Y is routed to a track $j > i+1$.

Thus, the clamp module assigned to net Y ensures that this net is routed in along its dedicated track i. It remains to show that net Y will not change the track afterwards. This is true for two reasons: first, horizontal and vertical capacity is not available, which is necessary for a change of tracks. Second, every time another net is routed in, this net is also combined with a clamp module such that a special track assignment is forced allowing no changes of the existing track assignment. This completes the proof of Proposition 5.2. □

We next argue that the track assignment does not change inside the gate and the detour module. For a change of tracks, vertical and horizontal capacity is needed, of course. Let us consider a gate module together with its two embedded detour modules. The free horizontal capacity of the detour modules is available from the left side of the left variable–line to the detour–line of the left detour module and from the detour–line of the right detour module to the left side of the right variable–line. The free horizontal capacity of the gate module is either available from the left side of the left detour–line to the column of maximum density inside the gate module or from this column to the right side of the right detour–line. The clamp module of the gate–net G_3 (or G_4) guarantees that at the vertical line where the free horizontal capacities overlap there is no vertical capacity available. For this reason, the capacities of the detour and the gate modules can be treated separately.

Lemma 5.3 (Detour Module). *For every layout of an instance of the switchbox routing problem, the track assignment of the nets does not change inside a detour module. In particular, the two new inserted detour–nets are assigned to the tracks of the corresponding ending detour–nets.*

Proof. Due to symmetry, we consider without loss of generality a right–handed detour module. Obviously, the vertical line on the left side of the variable–line is density–decreasing and the detour–line is density–increasing. Because of the terminal assignment of the detour module, this free horizontal capacity is always used by a net occupying two tracks. In case of a TRUE–routing, this is the detour–net e_{i-1}, and in case of a FALSE–routing, this is first the detour–net e_{i-1} and then the real–net whose terminal is assigned to the lower side of the variable–line (see Figure 5). Since all tracks are occupied by nets, changes of the track assignment are impossible.

To complete the proof, we show that the ordering of the track assignment of the starting detour–nets e_i and g_i corresponds to the ordering of the ending detour–nets e_{i-1} and g_{i-1}. Therefore, we consider the sandwich–net S_g whose left and right terminals are assigned to the sides of the switchbox between those of the upper detour–nets indexed with e and the lower detour–nets denoted by g. The arrangement at the right–most vertical line of the detour module together with the maximum density enforces that g_i must be routed below S_g (see Figure 5), since otherwise density would exaggerate capacity. Now, we suppose that the nets S_g and g_i are routed above e_i. As all new nets of the detour module start from the bottom of the switchbox, net S_g can only ascend in the track assignment. This means that at the right side of the switchbox the position of S_g in the track assignment is higher than the position of its last terminal. Due to the lack of free vertical capacity at the right side of the switchbox, net S_g cannot be connected with its right terminal. Thus, the track assignment is invariant inside the detour module. This completes the proof of this lemma. □

Lemma 5.4 (Gate Module). *For every layout of an instance of the resulting switchbox routing problem, the track assignment of the nets does not change inside a gate module.*

Proof. We show that a change of the track assignment within the gate module is contradicted by its special terminal assignment. Depending on the side where the detour of the autonomous interval of the gate module is routed, horizontal capacity is either available from the left side of the left detour–line to the column of maximum density inside the gate module or from this column to the right side of the right detour–line. Without loss of generality, we consider the case that the detour of the gate module is routed to the right. If a FALSE–routing is realized at the left detour–line, there is only horizontal capacity available. Consequently, no changes of tracks are possible, and we may restrict ourselves to the case of a TRUE–routing. Suppose that the free vertical capacity of the detour–line is used to re–route some net X to the track which was occupied before by the gate–net G_3. This means that at the next density–increasing vertical line, X lies below the starting gate–net G_2. At the following vertical line, gate–net

G_1 ends and G_2 occupies two tracks in order to lay out the detour of the gate module. Gate–net G_2 has to be joined at the next detour–line since otherwise the density would exaggerate the capacity at the next density–increasing vertical line (where the gate–net G_4 is inserted). At this vertical line, G_4 then cannot be assigned to the track to which it is forced to by its clamp module according to Proposition 5.2 because net X still occupies this track and it cannot be re–routed to another track. Hence, routing a net along the free track within the gate module leads to a contradiction to the assumption of the existence of a layout for the corresponding switchbox routing problem. □

Lemma 5.5 (Ordering Lemma). *For every layout of an instance of the switchbox routing problem, the upper variable–track of a variable $x_i \in \mathcal{X}$ is either occupied by the real–net X_i or by the real–net $\overline{X}_i$ inside the associated interval I^*. The vertical ordering of the real–nets X_i and $\overline{X}_i$ is the same within this interval.*

Proof. Consider the variable–line to which the left–most terminals of the real–nets X_i and $\overline{X}_i$ are assigned to. We assume without loss of generality that the left–most terminal of X_i is assigned to the lower side of this vertical line. Both dedicated variable–tracks are unoccupied which is guaranteed by the clamp modules assigned to the corresponding extension–nets as shown in Proposition 5.2. There are two major cases to be distinguished.

In case of a FALSE–routing, the real–nets have to pass each other which means that net X_i has to be routed above net $\overline{X}_i$. The only possibility for this is that even both real–nets are assigned to their dedicated tracks.

In case of a TRUE–routing, the situation is more complex. Again, there are two cases to be distinguished.

For the first case, we suppose that $\overline{X}_i$ occupies a track higher than its dedicated track by re–routing another net to the dedicated track. This means that $\overline{X}_i$ is assigned to a higher track than its corresponding sandwich–nets. Recall that the middle terminals of these sandwich–nets are assigned to the detour–lines of the detour modules which surround the variable–lines with terminals of X_i and $\overline{X}_i$ (see Figure 5). Hence, if a real–net is routed above its corresponding sandwich–nets, then a FALSE–routing cannot be realized for this variable. Since each variable occurs both negated as well as unnegated, for each pair of real–nets at least one FALSE–routing has to be realized. If once one real–net with index i lies above its corresponding sandwich–nets, then it can only be re-routed below its sandwich–nets when another real–net is routed above its corresponding sandwich–nets at the variable–line where its left–most terminal is assigned to. Consequently, there exists a pair of real–nets $(X_l, \overline{X}_l)$ with index $l \geq i$ for which a FALSE–routing cannot be realized. This contradicts the assumption on the existence of a layout.

For the second case, we suppose that both real–nets occupy tracks lower than the upper dedicated track. Then, we consider the vertical capacity at the right–most variable–line of x_i. It does not matter whether a TRUE– or a FALSE–routing has to be realized, the vertical capacity below the upper dedicated track is occupied by one of the associated real–nets. This implies that the net which occupies the upper dedicated track cannot be re–routed. But then, this gives a conflict with the clamp module of the corresponding upper extension–net $X_{i,rt}$. Hence, at least the upper dedicated track is occupied by one of the associated real–nets which proves the first part of this lemma.

Potentially, a change of the vertical ordering of a pair of real–nets can only happen inside the detour or the gate module using the capacity provided by these modules, or at variable–lines containing the left–most or right–most terminals of real–nets. From Lemmas 5.3 and 5.4 follows that the capacity of a detour or a gate module cannot be used for a change of tracks. Therefore, we may restrict our attention to variable–lines corresponding to left–most or right–most terminals of real–nets. First, we consider the left–most variable–line of a variable x_i. Recall that the variables (real–nets, respectively) are numbered in order of their first appearance in the clauses from the left to the right. This means that all terminals of the remaining real–nets $(X_l, \overline{X}_l)$, $l > i$, are placed to the right of the variable–line under consideration. As shown before, the real–net corresponding to x_i inserted from the top occupies at least the vertical capacity of the variable–line from the top to its upper dedicated track. This implies that from each pair of real–nets with index less than i — if they still exist at this point — at least the upper real–net remains on its track. Consequently, the ordering of each pair of those real–nets is maintained. Below the upper dedicated track of x_i, there are no pairs of real–nets. Furthermore, the vertical capacity of the corresponding detour–line cannot be used to change the ordering of a pair of real–nets since the critical capacity is occupied by a corresponding sandwich–net.

It remains to show that also at the right–most variable–line of a real–net the ordering of each pair of real–nets is preserved. In case of right–most variable–lines, there is no special numbering. We consider a pair of real–nets $(X_i,\overline{X}_i)$ and we assume without loss of generality that net X_i is routed along the upper dedicated track. Net $\overline{X}_i$ is routed below. Again, we distinguish two cases. First, we consider the right–most variable–line corresponding to a variable x_l, $l < i$. If at this variable–line a FALSE–routing has to be realized or if both real–nets X_l and $\overline{X}_l$ lie above net X_i, then obviously the ordering of the real–nets X_i and $\overline{X}_i$ cannot be changed since vertical capacity is not available for re–routing X_i or $\overline{X}_i$. Therefore, we assume that the real–nets X_i and $\overline{X}_i$ lie between X_l and $\overline{X}_l$. This means that the lower dedicated track of the variable x_l is occupied by a net Y. This net has to be re–routed to a lower track. Otherwise, there is a contradiction to the clamp module of the extension–net $X_{l,rb}$. This means that the real–net $\overline{X}_i$ cannot be re–routed to a higher track than the associated real–net X_i. On the other hand, the upper real–net X_i cannot be re–routed below net $\overline{X}_i$ since then, both nets lie below their associated upper dedicated track. Since at right–most variable–lines nets can only descend in the track assignment, this gives a conflict with the clamp module of their associated extension–nets. Thus, the ordering of the real–nets X_i and $\overline{X}_i$ is preserved in this case.

For the second case, we consider the right–most variable–line corresponding to a variable x_l, $l > i$. But in this case, the vertical capacity of the variable–line is used at least from the upper side of the switchbox to the upper dedicated track of the variable x_l. Moreover, the vertical capacity of the detour–line is used in this area by a corresponding sandwich–net. This implies that the ordering of the real–nets X_i and $\overline{X}_i$ is also maintained inside the associated interval I^*. This proves this lemma. The restriction of the definition of I^* takes the following situation into account. At the columns inside the corresponding detour modules between the endpoints of I^* and the left–most (right–most, respectively) variable–line of the associated real–nets the ordering may be contrary to the ordering in I^* since the top real–net may not use the upper dedicated track anymore. Then, it is routed along the detour–line and the horizontal capacity of the detour module. Thereby, it is routed below its complementary real–net. But in this case, the detour of the assigned gate module cannot be realized at this detour–line, which means that there exists another literal which satisfies the clause. The unoccupied dedicated track cannot be used by any other net since the clamp module of the associated upper extension–net follows at the next vertical lines.

This completes the proof of Lemma 5.5. □

Lemma 5.6 (Gate Lemma). *Consider a layout for the resulting instance of the switchbox routing problem. Then, in every clause block, a* TRUE*–routing is realized for at least one variable of the associated clause.*

Proof. As mentioned before, a clause block corresponding to a 3–literal (2–literal) clause consists of 4(2) detour–lines. At the detour–lines, vertical capacity is available in order to route a detour of a gate module or to realize a FALSE–routing. A detour of a gate module and a FALSE–routing cannot be realized at the same detour–line since FALSE–routings occupy the vertical capacity of the detour–lines from the area of the detour–nets to the area of the extension/real–nets. Hence, a FALSE–routing occupies the area where the detours of the gate modules have to be routed (see Figure 8).

Suppose a layout of the switchbox routing problem implies a variable assignment which does not satisfy a clause C. Hence, at each variable–line of the clause block corresponding to C a FALSE–routing is realized. This means that for a 3–literal (2–literal) clause, 3(2) FALSE–routings plus 2 (1) detours of the gate modules have to be realized. Thus, the demand of vertical capacity is strictly greater than the number of detour–lines which contradicts the existence of a layout. □

Proof of Theorem 2.1. Simple counting shows that all introduced nets consist of at most three terminals. Then, the theorem follows directly from the above lemmas. Lemma 5.1 states that a satisfying variable assignment induces a layout of the corresponding instance of the switchbox routing problem. Such a layout can be constructed by the algorithm which is given in the proof of Lemma 5.1. For the other direction, the variable assignment is shown to be well defined by Lemma 5.5 while Lemma 5.6 guarantees that every clause of the 3–bounded 3–SAT instance is satisfied by the obtained variable assignment. Hence, the 3–terminal switchbox routing problem is NP–complete. □

6 Concluding Remarks

In this paper, we found the border between the polynomial time solvable and the NP–complete instances of the switchbox routing problem. The transition between the 2–terminal case (which can be solved efficiently) and the k–terminal case, $k \geq 3$, (which is NP–complete) corresponds directly to the transition between path embeddings and Steiner tree embeddings in grid graphs. The techniques introduced in this paper also apply to the channel routing problem. A proof similar to the one above shows that the 5–terminal channel routing problem is NP–complete [Har96]. This improves upon the result of [Sar87] for 6–terminal nets. The complexity status of the 3– and 4–terminal channel routing problem remains open.

Acknowledgments

The authors are grateful to Dorothea Wagner for helpful comments and discussions.

References

[FWW93] M. Formann, D. Wagner, and F. Wagner. Routing through a dense channel with minimum total wire length. *Journal of Algorithms*, 15:267 – 283, 1993.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability – A Guide to the Theory of NP–Completeness*. W. H. Freeman and Company, New York, 1979.

[GMW93] M. Grötschel, A. Martin, and R. Weismantel. Routing in grid graphs by cutting planes. In G. Rinaldi and L. Wolsey, editors, *Third IPCO Conference*, pages 447 – 461, 1993.

[Har96] S. Hartmann. On the NP –completeness of channel and switchbox routing problems. Preprint no. 542–1996, Fachbereich Mathematik, Technische Universität Berlin, Berlin, Germany, 1996.

[KPS90] B. Korte, H.-J. Prömel, and A. Steger. Steiner trees in VLSI–layout. In B. Korte, L. Lovasz, H.-J. Prömel, and A. Schrijver, editors, *Paths, Flows and VLSI–Layout*, pages 185–214. Springer Verlag, 1990.

[KvL84] M. R. Kramer and J. van Leeuwen. The complexity of wire routing and finding minimum area layouts for arbitrary VLSI circuits. In F. P. Preparata, editor, *Advances in Computing Research, Vol. 2 VLSI theory*, pages 129–146. JAI Press, Reading, MA, 1984.

[Len90] Th. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout*. Teubner/Wiley&Sons, 1990.

[Mid93] M. Middendorf. Manhattan channel routing is NP–complete under truly restricted settings. Preprint, Universität Karlruhe, 1993. To appear in the Chicago Journal of Theoretical Computer Science.

[MP86] K. Mehlhorn and F. P. Preparata. Routing through a rectangle. *Journal of the Association for Computing Machinery*, 33:60 – 85, 1986.

[MS92] M. Marek-Sadowska. Switch box routing: a retrospective. *Integration, the VLSI Journal*, 13:39 – 65, 1992.

[MWW95] R. H. Möhring, D. Wagner, and F. Wagner. VLSI network design. In M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors, *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*, chapter 8, pages 625 – 712. Elsevier, 1995.

[OS81] H. Okamura and P. D. Seymour. Multicommodity flows in planar graphs. *Journal of Computer Theory*, 31:75 – 81, 1981.

[Sar87] M. Sarrafzadeh. Channel–routing problem in the knock–knee mode is NP–complete. *IEEE Transaction on Computer–Aided Design*, 6:503 – 506, 1987.

[Szy85] T. G. Szymanski. Dogleg channel–routing is NP–complete. *IEEE Transaction on Computer–Aided Design*, 4:31 – 41, 1985.