ON SHIFTING BLOCKS AND TERMINALS TO MINIMIZE CHANNEL DENSITY

Yang Cai and D. F. Wong

Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712-1188

TR-92-08 March 1992

On Shifting Blocks and Terminals to Minimize Channel Density [*†]

Yang Cai and D. F. Wong
Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712

Abstract

We study in this paper the problem of minimizing channel density by simultaneously shifting the blocks that form the two sides of a channel and the terminals on the boundary of each block. Several special cases of this problem have been investigated, but no optimal algorithm was known for the general case. We present an optimal algorithm for solving this problem. For long channels, we also propose effective heuristic techniques to speed up our algorithm. Extensions as well as aplications of our algorithms to detailed routing in building-block layout design are also discussed.

1 Introduction

The channel routing problem plays an important role in the physical design of VLSI circuits and has been extensively studied in the past [8]. Conventional channel routers assume all terminals (pins) on the two sides of the channel have fixed positions. However, it is typical in practice that the exact positions of the terminals are not completely fixed at the beginning of the routing phase [1, 2, 3, 5, 6, 7]. The existence of floating terminals can be used to our advantage to obtain further area reduction.

In this paper, we consider the problem of minimizing channel density by shifting blocks and terminals. Each side of the channel consists of a row of blocks, and the terminals are

*This work was partially supported by the National Science Foundation under grant MIP-8909586, by an IBM Faculty Development Award, and by an ACM SIGDA scholarship.

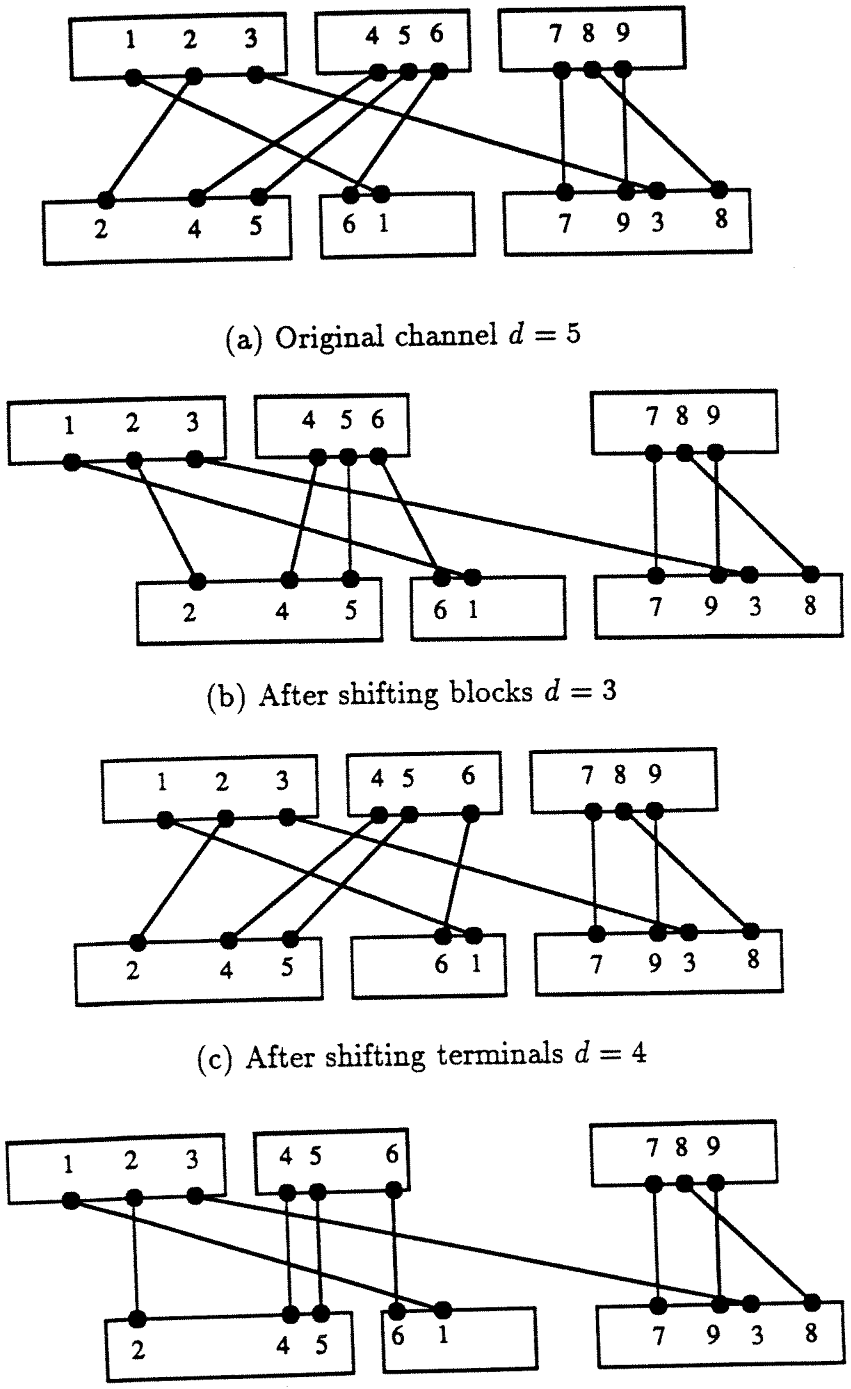
†A preliminary version of this paper was presented at ICCAD-91 [4].

located on the boundaries of these blocks, as illustrated in Figure 1(a). The relative ordering of the blocks on each side of the channel, as well as the relative ordering of the terminals on the boundary of each block are fixed. The problem is to laterally shift the blocks and the terminals, subject to the constraint that the terminals stay in the blocks they belong to, such that the density of the resulting channel is minimized.

No optimal algorithm was previously known for this problem, although optimal algorithms for several special cases of this problem have been developed before. In [7], the authors considered the case where each side of the channel consists of exactly one block, and only the blocks are allowed to be shifted. This was generalized in [3, 5] to allow multiple blocks on each side of the channel under the formulation of a channel pin assignment problem. Another special case was considered in [6]. Again each side of the channel consists of exactly one block, but this time only the terminals are allowed to be shifted, not the blocks. The restriction that each side of the channel consists of exactly one block was relaxed again in [3, 5]. Note that the channel pin assignment model presented in [3, 5] cannot be used to formulate the general problem of shifting blocks and terminals to minimize channel density.

Figure 1 shows that by allowing the blocks and the terminals both to be shifted, more reduction in channel density can be achieved than by allowing only the terminals or only the blocks to be shifted. For the original channel shown in Figure 1(a), the minimum density achievable by shifting only the blocks is 3, as shown in Figure 1(b); the minimum density achievable by shifting only the terminals inside each block is 4, as shown in Figure 1(c). However, if the blocks and the terminals are allowed to be shifted, we can achieve minimum density 2, as shown in Figure 1(d).

The remainder of this paper is organized as follows. Preliminaries, including terminologies and notations are given in Section 2. The optimal algorithm is then presented in Section 3. Section 4 extends the algorithm to handle channels with exits, irregular boundaries, and independent terminals, and to handle additional pin/block assignment constraints. Section 5 proposes fast heuristic approaches for speeding up the algorithm. Section 6 presents some preliminary experimental results. Section 7 discusses applications of our algorithms. Finally, Section 8 concludes the paper with some remarks.



(a) Original channel $d = 5$

(b) After shifting blocks $d = 3$

(c) After shifting terminals $d = 4$

(d) After shifting blocks and terminals $d = 2$

Figure 1: Minimizing channel density by shifting blocks and terminals

2 Preliminaries

In this section we present some useful terminologies and preliminary results.

2.1 Basic Terminologies

Given a channel with length L, as before let $TOP = \{t_1, t_2, \ldots, t_p\}$ and $BOTTOM = \{b_1, b_2, \ldots, b_q\}$ be the set of terminals on the top and bottom of the channel (from left to right in that order), respectively, where $p = |TOP|$ and $q = |BOTTOM|$. The set of nets $N = \{N_1, N_2, \ldots, N_n\}$ connecting these terminals is a partition of $TOP \cup BOTTOM$, such that $N_i \subseteq TOP \cup BOTTOM$ contains the set of terminals of the ith net, $1 \leq i \leq n$. Let $T_1, T_2, \ldots, T_a$ and $B_1, B_2, \ldots, B_b$ be the sets of blocks that form the top and bottom side of the channel (from left to right in that order), respectively. The length of the block containing terminal t_i is denoted by l_i, $1 \leq i \leq p$, and the length of the block containing terminal b_j is denoted by l'_j, $1 \leq j \leq q$.

For convenience of presentation, we introduce a terminal at each corner of a block, which by itself forms a trivial net. These terminals are called *corner terminals* and their positions inside the blocks are fixed (at the corners of the blocks). The positions of the corner terminals will be used to determine the positions of the blocks. We also make use of two Boolean functions *left_corner* and *right_corner*, such that for any terminal t, left_corner(t) = **true** if and only if t is a terminal located at the left corner of a block. Similarly, right_corner(t) = **true** if and only if t is a terminal located at the right corner of a block. It is convenient to introduce an auxiliary column 0, and two additional terminals t_0 and b_0, each by itself forms of a trivial net. We also require that t_0 (b_0) be the leftmost terminal on the top (bottom) of the channel.

A channel is said to be *derivable* from another channel if and only if it can be obtained from that channel by shifting blocks and terminals. Hence the problem we consider is to compute a minimum density channel derivable from a given channel.

2.2 (i, j, k, u, v)-Channels

A channel is said to be an (i, j, k, u, v)-*channel* if and only if

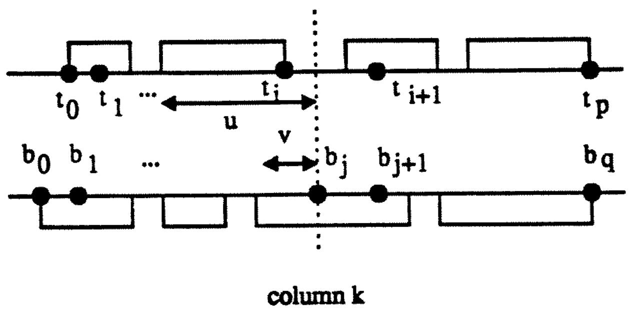


Figure 2: An (i, j, k, u, v)-channel

1. $t_0, t_1, \ldots, t_i, b_0, b_1, \ldots, b_j$ are the only terminals on the first k columns of the channel;
2. The left corner of the block containing terminal t_i is on the $(k-u)$th column of the channel; and
3. The left corner of the block containing terminal b_j is on the $(k-v)$th column of the channel.

This is illustrated in Figure 2. Note that the last two arguments are introduced to make sure that no terminal is shifted out of the block it belongs to. We can further classify the (i, j, k, u, v)-channels into the following four types, according to how the two endpoints of column k are occupied by terminals.

- **Type 0:** No terminal is on column k;
- **Type 1:** Only t_i is on column k;
- **Type 2:** Only b_j is on column k;
- **Type 3:** Both t_i and b_j are on column k.

Lemma 2.1 *All (i, j, k, u, v)-channels of the same type which are derivable from the same channel have the same local density at column k.*

Proof: The local density at column k of a channel only depends on how the terminals are distributed with respected to column k, *i.e.*, which terminals are located to the left of column k, which terminals are located on column k, and which terminals are located to the right of column k. Since all (i, j, k, u, v)-channels of the same type which are derivable from the same channel have the same distribution of nets which respect to column k, their local densities at column k are equal. □

2.3 Crossing Numbers

The common local density of an (i, j, k, u, v)-channel of type 1 derivable from a given channel is equal to the number of nets with one terminal in

$$\{t_1, t_2, \ldots, t_{i-1}, b_1, b_2, \ldots, b_j\},$$

and one terminal in

$$TOP \cup BOTTOM - \{t_1, t_2, \ldots, t_i, b_1, b_2, \ldots, b_j\}$$

plus δ, where $\delta = 1$ if the net containing terminal t_i is not a trivial net, and it has not been accounted for already, otherwise it is 0. Clearly, this number is independent of the values of k, u and v, and it is well defined even if there is no (i, j, k, u, v)-channel of type 1 derivable from the given channel exists. We will denote this number by $R_1(i, j)$. If there exists an (i, j, k, u, v)-channel of type 1 derivable from the given channel, then $R_1(i, j)$ is equal to the common local density at column k of all (i, j, k, u, v)-channel of type 1 derivable from the given channel, otherwise it is just a number with no meaning associated with it. We can similarly define the numbers $R_0(i, j)$, $R_2(i, j)$ and $R_3(i, j)$. The number $R_h(i, j,)$ is referred to as the *crossing number of type h at (i, j)*, $0 \leq h \leq 3$. The crossing numbers at (i, j) is similar to the crossing numbers at (i, j, k) as defined in [3, 5], except that position constraints are not considered in this case. In much the same way as computing the k-densities for a given channel, the crossing numbers can be computed in $O(pqL)$ time for all $0 \leq i \leq p$, $0 \leq j \leq q$.

2.4 Generalized Crossing Numbers

We now introduce the *generalized crossing numbers*, which play an important role in our algorithm. Intuitively, the generalized crossing number of type h $(0 \leq h \leq 3)$ at (i, j, k, u, v) is equal to $+\infty$ if we can detect "easily" at column k that the given channel has no (i, j, k, u, v)-channel derivable from it, otherwise it is equal to the crossing number of type h at (i, j), *i.e.*,

$R_h(i,j)$. Therefore, if the given channel has an (i,j,k,u,v)-channel of type h derivable from it, then the generalized crossing number of type h at (i,j,k,u,v) is equal to the local density at column k of any (i,j,k,u,v)-channel derivable from the given channel of type h. Besides this, the generalized crossing numbers also help us to detect if the given channel has no derivable channel of a specific form. The precise definition of the generalized crossing numbers is given by Procedure Generalized_Crossing_Numbers. For each 5-tuple (i,j,k,u,v), Procedure Generalized_Crossing_Numbers computes four numbers $w(i,j,k,u,v)$, $x(i,j,k,u,v)$, $y(i,j,k,u,v)$, and $z(i,j,k,u,v)$, which are referred to as the *generalized crossing number of type 0, type 1, type 2, type 3*, respectively, *at* (i,j,k,u,v). [1]

```
Procedure: Generalized_Crossing_Numbers ($C$);
  (* $C$ is a given channel *)
  begin
    for $(i,j,k,u,v) := (0,0,0,0,0)$ to $(p,q,L,L,L)$ do
      begin
        if $(u = 0)$ or $(v = 0)$ or $(\max\{u,v\} > k)$
          then $w(i,j,k,u,v) := +\infty$
          else $w(i,j,k,u,v) := R_0(i,j)$;
        if (right_corner($t_i$) and $u \neq l_i$) or $(\max\{u,v\} > k)$ or
          (left_corner($t_i$) and $u \neq 0$) or
          (not (left_corner($t_i$)) and $u = 0$) or $(v = 0)$
          then $x(i,j,k,u,v) := +\infty$
          else $x(i,j,k,u,v) := R_1(i,j)$;
        if (right_corner($b_j$) and $v \neq l'_j$) or $(\max\{u,v\} > k)$ or
          (left_corner($b_j$) and $v \neq 0$) or
          (not (left_corner($b_j$)) and $v = 0$) or $(u = 0)$
          then $y(i,j,k,u,v) := +\infty$
          else $y(i,j,k,u,v) := R_2(i,j)$;
        if (right_corner($t_i$) and $u \neq l_i$) or (right_corner($b_j$) and $v \neq l'_j$) or
          $(\max\{u,v\} > k)$ or
          (left_corner($t_i$) and $u \neq 0$) or (left_corner($b_j$) and $v \neq 0$) or
          (not (left_corner($t_i$)) and $u = 0$) or (not (left_corner($b_j$)) and $v = 0$)
          then $z(i,j,k,u,v) := +\infty$
          else $z(i,j,k,u,v) := R_3(i,j)$
```

[1]We use "**for** $(i_1, i_2, \ldots, i_m) := (a_1, a_2, \ldots, a_m)$ **to** $(b_1, b_2, \ldots, b_m)$ **do**" as a shorthand for

```
for $i_1 := a_1$ to $b_1$ do
  for $i_2 := a_2$ to $b_2$ do
    ......
      for $i_m := a_m$ to $b_m$ do.
```

```
        end
    end;
```

The properties of the generalized crossing numbers are stated in the following lemmas, whose proofs are immediate once we observe that whenever a generalized crossing number is set to $+\infty$ by Procedure Generalized_Crossing_Numbers, there is no (i, j, k, u, v)-channel of the corresponding form derivable from the given channel.

Lemma 2.2 *Given a channel, if there exists an (i, j, k, u, v)-channel of type 0 (type 1, type 2, type 3, respectively) derivable from it, then $w(i, j, k, u, v)$ $(x(i, j, k, u, v)$, $y(i, j, k, u, v)$, $z(i, j, k, u, v)$, respectively) equals the common local density at column k of all (i, j, k, u, v)-channels of type 0 (type 1, type 2, type 3, respectively) derivable from the given channel.*

Lemma 2.3 *Given a channel, if $w(i, j, k, u, v)$ $(x(i, j, k, u, v)$, $y(i, j, k, u, v)$, $z(i, j, k, u, v)$, respectively) $= +\infty$, then there is no (i, j, k, u, v)-channel of type 0 (type 1, type 2, type 3, respectively) derivable from the given channel.*

It should be noted that the converse of Lemma 2.3 is not true, *i.e.*, it is possible that $w(i, j, k, u, v)$ $(x(i, j, k, u, v)$, $y(i, j, k, u, v)$, $z(i, j, k, u, v)$, respectively) $< +\infty$ and yet the given channel has no derivable (i, j, k, u, v)-channel of type 0 (type 1, type 2, type 3, respectively). In this case, there is no meaning associated with $w(i, j, k, u, v)$ $(x(i, j, k, u, v)$, $y(i, j, k, u, v)$, $z(i, j, k, u, v)$, respectively). This is deliberately done in order to facilitate the design of our algorithm. We conclude this section by stating the following lemma.

Lemma 2.4 *All the generalized crossing numbers of a given channel can be computed in $O(pqL^3)$ time and space.*

Proof: The crossing numbers can be computed in $O(pqL)$ time. Once the values of the crossing numbers are computed, each generalized crossing number can be computed in constant time according to Procedure Generalized_Crossing_Numbers. Hence the lemma follows. □

3 The Optimal Algorithm

We present in this section our optimal algorithm. Section 3.1 introduces the concept of a density function of a given channel. Our algorithm is based on the computation of a density function. Section 3.2 presents the optimal algorithm.

3.1 Density Function

Our algorithm first computes a density function f, then constructs a channel derivable from the given channel with minimum density based on f. Ideally, we would like to define the density function f of a given channel to be the unique function whose value at (i, j, k, u, v), $0 \leq i \leq p$, $0 \leq j \leq q$, $0 \leq k, u, v \leq L$, is equal to the minimum k-density of any (i, j, k, u, v)-channel derivable from the given channel, so that $f(i, j, k, u, v) = +\infty$ if the given channel has no derivable (i, j, k, u, v)-channel. However, this simplistic definition renders the computation of a density function intractable. We avoid this difficulty by defining a *density function* with respect to the given channel to be a function f satisfying the following conditions:

1. For all $0 \leq i \leq p$, $0 \leq j \leq q$, $0 \leq k, u, v \leq L$, if the given channel has an (i, j, k, u, v)-channel derivable from it, then $f(i, j, k, u, v)$ is equal to the minimum k-density of any (i, j, k, u, v)-channel derivable from it;

2. For all $0 \leq k, u, v \leq L$, if the given channel has no (p, q, k, u, v)-channel derivable from it, then $f(p, q, k, u, v) = +\infty$.

According to this definition, it is no longer true that for all i, j, $1 \leq i \leq p$, $1 \leq j \leq q$, the given channel has no (i, j, k, u, v)-channel derivable from it implies $f(i, j, k, u, v) = +\infty$. However, Condition 2 is still strong enough to guarantee the correctness of our algorithm. The following lemma follows directly from the definition of a density function.

Lemma 3.1 *Let f be a density function of a given channel, then*

$$D = \min_{0 \leq u, v \leq L} f(p, q, L, u, v) < +\infty$$

is the minimum density of any channel derivable from the given channel.

To simplify the presentation of our algorithm, we define four intermediate functions d_0, d_1, d_2 and d_3 in terms of a density function f of a given channel. For $0 \leq i \leq p$, $0 \leq j \leq q$, $0 \leq k, u, v \leq L$, we define $d_0(i, j, k, u, v)$, $d_1(i, j, k, u, v)$, $d_2(i, j, k, u, v)$ and $d_3(i, j, k, u, v)$ as follows. (For ease of exposition, we assume $0 - 1 = 0$ in the rest of this paper.)

$$\begin{aligned}
d_0(i,j,k,u,v) &= \min\{\max\{f(i,j,k-1,u-1,v-1), w(i,j,k,u,v)\},\\
&\qquad \max\{f(i-1,j,k-1,u-1,v-1), x(i,j,k,u,v)\},\\
&\qquad \max\{f(i,j-1,k-1,u-1,v-1), y(i,j,k,u,v)\},\\
&\qquad \max\{f(i-1,j-1,k-1,u-1,v-1), z(i,j,k,u,v)\}\},
\end{aligned}$$

$$\begin{aligned}
d_1(i,j,k,u,v) &= \min\{\max\{f(i,j,k-1,u-1,v-1), w(i,j,k,u,v)\},\\
&\qquad \max\{\min_{0\le u'<k} f(i-1,j,k-1,u',v-1), x(i,j,k,u,v)\},\\
&\qquad \max\{f(i,j-1,k-1,u-1,v-1), y(i,j,k,u,v)\},\\
&\qquad \max\{\min_{0\le u'<k} f(i-1,j-1,k-1,u',v-1), z(i,j,k,u,v)\}\},
\end{aligned}$$

$$\begin{aligned}
d_2(i,j,k,u,v) &= \min\{\max\{f(i,j,k-1,u-1,v-1), w(i,j,k,u,v)\},\\
&\qquad \max\{f(i-1,j,k-1,u-1,v-1), x(i,j,k,u,v)\},\\
&\qquad \max\{\min_{0\le v'<k} f(i,j-1,k-1,u-1,v'), y(i,j,k,u,v)\},\\
&\qquad \max\{\min_{0\le v'<k} f(i-1,j-1,k-1,u-1,v'), z(i,j,k,u,v)\}\},
\end{aligned}$$

$$\begin{aligned}
d_3(i,j,k,u,v) &= \min\{\max\{f(i,j,k-1,u-1,v-1), w(i,j,k,u,v)\},\\
&\qquad \max\{\min_{0\le u'<k} f(i-1,j,k-1,u',v-1), x(i,j,k,u,v)\},\\
&\qquad \max\{\min_{0\le v'<k} f(i,j-1,k-1,u-1,v'), y(i,j,k,u,v)\},\\
&\qquad \max\{\min_{0\le u',v'<k} f(i-1,j-1,k-1,u',v'), z(i,j,k,u,v)\}\}.
\end{aligned}$$

The physical meanings of $d_0(i,j,k,u,v)$, $d_1(i,j,k,u,v)$, $d_2(i,j,k,u,v)$ and $d_3(i,j,k,u,v)$ are related to the following four cases, depending on whether t_i or b_j is a left corner terminal.

- **Case 0:** Neither t_i nor b_j is a left corner terminal;
- **Case 1:** Only t_i is a left corner terminal;
- **Case 2:** Only b_j is a left corner terminal;
- **Case 3:** Both t_i and b_j are left corner terminals.

More specifically, we have

Lemma 3.2 *Let f be a density function of a given channel. If there exists an (i,j,k,u,v)-channel derivable from the given channel such that (t_i, b_j) is of Case h, $0 \leq h \leq 3$, then $d_h(i,j,k,u,v)$ is equal to the minimum k-density of any (i,j,k,u,v)-channel derivable from the given channel where (t_i, b_j) is of Case h.*

Proof: Suppose there exists an (i,j,k,u,v)-channel derivable from the given channel and (t_i, b_j) is of Case 2. An (i,j,k,u,v)-channel derivable from the given channel with minimum k-density is either of type 0, type 1, type 2 or type 3. If it is of type 0, then its k-density is equal to the maximal of its local density at column k, which is $w(i,j,k,u,v)$ by Lemma 3.2, and the minimum $(k-1)$-density of any $(i,j,k-1,u-1,v-1)$-channel derivable from the given channel, which is $f(i,j,k-1,u-1,v-1)$ by the inductive hypothesis. Therefore, in this case the k-density of this channel is given by

$$\max\{f(i,j,k-1,u-1,v-1), w(i,j,k,u,v)\}.$$

If it is of type 2, then its k-density is equal to the maximal of its local density at column k, which is $y(i,j,k,u,v)$, and the minimum $(k-1)$-density of any $(i,j-1,k-1,u-1,v')$-channel derivable from the given channel with $0 \leq v' < k$, which is $\min\{f(i,j-1,k-1,u-1,v') : 0 \leq v' < k\}$ by the inductive hypothesis. (Since b_j is a left corner terminal, terminal b_{j-1} belongs to a new block, and the position of this block is not known. We have the freedom of choosing a position for it so that the $(k-1)$-density of the channel under consideration is minimized.) Hence in this case the k-density of this channel is given by

$$\max\{\min_{0 \leq v' < k} f(i,j-1,k-1,u-1,v'), y(i,j,k,u,v)\}.$$

Similar analysis shows that if the channel is of type 1, then its k-density is given by

$$\max\{f(i-1,j,k-1,u-1,v-1), x(i,j,k,u,v)\},$$

and if it is of type 3, then its k-density is given by

$$\max\{\min_{0 \leq v' < k} f(i-1,j-1,k-1,u-1,v'), z(i,j,k,u,v)\}.$$

Hence $d_2(i,j,k,u,v)$ is equal to the minimum k-density of any (i,j,k,u,v)-channel derivable from the given channel if (t_i, b_j) is of Case 2. The proofs of the other three cases are similar to that of Case 2. □

Although it appears that in the worst case we need $O(L)$ time to compute $d_1(i,j,k,u,v)$, $d_2(i,j,k,u,v)$, and $O(L^2)$ time to compute $d_3(i,j,k,u,v)$, we can reduce the computation time to $O(1)$ by introducing the following auxiliary functions.

For $0 \le i \le p$, $0 \le j \le q$, $0 \le k,u,v \le L$, define

$$\begin{aligned}
f_1(i,j,k,u,v) &= \min_{0 \le u' \le u} f(i,j,k,u',v); \\
f_2(i,j,k,u,v) &= \min_{0 \le v' \le v} f(i,j,k,u,v'); \\
f_3(i,j,k,u,v) &= \min_{0 \le u' \le u, 0 \le v' \le v} f(i,j,k,u',v').
\end{aligned}$$

Obviously, we have

$$\begin{aligned}
f_1(i,j,k,0,v) &= f(i,j,k,0,v); \\
f_2(i,j,k,u,0) &= f(i,j,k,u,0); \\
f_3(i,j,k,0,v) &= f_2(i,j,k,0,v); \\
f_3(i,j,k,u,0) &= f_1(i,j,k,u,0).
\end{aligned}$$

In general, these functions can be computed recursively according to the following formulas:

$$\begin{aligned}
f_1(i,j,k,u,v) &= \min\{f_1(i,j,k,u-1,v), f(i,j,k,u,v)\}; \\
f_2(i,j,k,u,v) &= \min\{f_2(i,j,k,u,v-1), f(i,j,k,u,v)\}; \\
f_3(i,j,k,u,v) &= \min\{f_3(i,j,k,u-1,v), f_2(i,j,k,u,v)\} \\
&= \min\{f_3(i,j,k,u,v-1), f_1(i,j,k,u,v)\}.
\end{aligned}$$

The formulas for d_1, d_2, and d_3 can now be rewritten as follows:

$$\begin{aligned}
d_1(i,j,k,u,v) = \min\{&\max\{f(i,j,k-1,u-1,v-1), w(i,j,k,u,v)\}, \\
&\max\{f_1(i-1,j,k-1,k-1,v-1), x(i,j,k,u,v)\}, \\
&\max\{f(i,j-1,k-1,u-1,v-1), y(i,j,k,u,v)\}, \\
&\max\{f_1(i-1,j-1,k-1,k-1,v-1), z(i,j,k,u,v)\}\};
\end{aligned}$$

$$\begin{aligned}
d_2(i,j,k,u,v) = \min\{&\max\{f(i,j,k-1,u-1,v-1), w(i,j,k,u,v)\}, \\
&\max\{f(i-1,j,k-1,u-1,v-1), x(i,j,k,u,v)\}, \\
&\max\{f_2(i,j-1,k-1,u-1,k-1), y(i,j,k,u,v)\},
\end{aligned}$$

$$\max\{f_2(i-1, j-1, k-1, u-1, k-1), z(i,j,k,u,v)\}\};$$

$$\begin{aligned} d_3(i,j,k,u,v) \quad = \quad \min\{&\max\{f(i,j,k-1,u-1,v-1), w(i,j,k,u,v)\}, \\ &\max\{f_1(i-1,j,k-1,k-1,v-1), x(i,j,k,u,v)\}, \\ &\max\{f_2(i,j-1,k-1,u-1,k-1), y(i,j,k,u,v)\}, \\ &\max\{f_3(i-1,j-1,k-1,k-1,k-1), z(i,j,k,u,v)\}\}. \end{aligned}$$

It is now clear that the value of $d_1(i,j,k,u,v)$, $d_2(i,j,k,u,v)$, $d_3(i,j,k,u,v)$ can be computed in $O(1)$ time from the previously computed values.

3.2 The Algorithm

We are now ready to present our algorithm. The algorithm first computes a density function f based on the formulas presented in the last subsection using dynamic programming, then it calls Procedure Generalized_Crossing_Numbers to construct an optimal solution by backtracking.

```
Algorithm: Minimizing_Channel_Density ($C$);
  (* $C$ is a given channel *)
  Begin
    (* initialize *)
    for $(i,j,u,v) := (0,0,0,0)$ to $(p,q,L,L)$ do
      $f(i,j,0,u,v) := +\infty$;
    for $(k,u,v) := (0,0,0)$ to $(L,L,L)$ do
      if $k \geq \max\{u,v\}$
        then $f(0,0,k,u,v) := 0$
        else $f(0,0,k,u,v) := +\infty$;
    (* compute a density function $f$ using dynamic programming *)
    for $(k,i,j,u,v) := (1,0,0,0,0)$ to $(L,p,q,L,L)$ do
      if left_corner($t_i$) and left_corner($b_j$)
        then $f(i,j,k,u,v) := d_3(i,j,k,u,v)$
        else if left_corner($b_j$)
              then $f(i,j,k,u,v) := d_2(i,j,k,u,v)$
              else if left_corner($t_i$)
                    then $f(i,j,k,u,v) := d_1(i,j,k,u,v)$
                    else $f(i,j,k,u,v) := d_0(i,j,k,u,v)$;
    (* construct an optimal solution using backtracking *)
    Call Procedure Construct_Optimal_Channel to construct an optimal channel
  End.
```

Given a density function f of the given channel, Procedure Construct_Optimal_Channel determines a position for each block (the position of a block is determined by those of its corner terminals) and a position for each terminal on its block based on the values of f, such that the resulting channel has minimum density among all channels derivable from the given channel.

```
Procedure: Construct_Optimal_Channel ($C$);
  (* $C$ is a given channel *)
  Begin
    $i := p; j := q;$
    Compute $u$, $v$, such that $f(p,q,L,u,v) = \min\{f(p,q,L,u',v') : 0 \leq u',v' \leq L\}$;
    for $k := L$ downto 1 do
      case $f(i,j,k,u,v)$ of
        $\max\{f(i,j,k-1,u-1,v-1), w(i,j,k,u,v)\}$ :
          $u := u-1;\ v := v-1;$
        $\max\{f(i-1,j,k-1,u',v-1), x(i,j,k,u,v)\}$ :
          Assign $t_i$ to column $k$;
          $i := i-1;\ u := u';\ v := v-1;$
        $\max\{f(i,j-1,k-1,u-1,v'), y(i,j,k,u,v)\}$ :
          Assign $b_j$ to column $k$;
          $j := j-1;\ u := u-1;\ v := v';$
        $\max\{f(i-1,j-1,k-1,u',v'), z(i,j,k,u,v)\}$ :
          Assign $t_i$, $b_j$ to column $k$;
          $i := i-1;\ j := j-1;\ u := u';\ v := v';$
        $\max\{f(i-1,j-1,k-1,u-1,v-1), z(i,j,k,u,v)\}$ :
          Assign $t_i$, $b_j$ to column $k$;
          $i := i-1;\ j := j-1;\ u := u-1;\ v := v-1;$
        $\max\{f(i-1,j,k-1,u-1,v-1), x(i,j,k,u,v)\}$ :
          Assign $t_i$ to column $k$;
          $i := i-1;\ u := u-1;\ v := v-1;$
        $\max\{f(i,j-1,k-1,u-1,v-1), y(i,j,k,u,v)\}$ :
          Assign $b_j$ to column $k$;
          $j := j-1;\ u := u-1;\ v := v-1;$
        $\max\{f(i-1,j-1,k-1,u',v-1), z(i,j,k,u,v)\}$ :
          Assign $t_i$, $b_j$ to column $k$;
          $i := i-1;\ j := j-1;\ u := u';\ v := v-1;$
        $\max\{f(i-1,j-1,k-1,u-1,v'), z(i,j,k,u,v)\}$ :
          Assign $t_i$, $b_j$ to column $k$;
          $i := i-1;\ j := j-1;\ u := u-1;\ v := v'$
      end
  End.
```

Note that we can also incorporated heuristics into Procedure Construct_Optimal_Channel to avoid introducing cycles or long paths in the vertical constraint digraph of the resulting channel.

Theorem 3.3 *The function f computed by Algorithm Minimizing_Channel_Density is a density function of the given channel.*

The proof of Theorem 3.3 is based on induction on the column number k that the function f satisfies the two conditions of being a density function of the given channel. We can now state the correctness of Algorithm Minimizing_Channel_Density.

Theorem 3.4 *Algorithm Minimizing_Channel_Density correctly constructs a channel derivable from the given channel with minimum density in $O(pqL^3)$ time and space.*

Proof: The correctness of the algorithm follows from the above discussions. We have seen earlier that the functions d_0, d_1, d_2 and d_3 can be computed in $O(pqL^3)$. Since each step of the for loop in Procedure Construct_Optimal_Channel can be done in $O(L^2)$ time, the whole procedure takes $O(L^3)$. Hence the overall complexity of the algorithm is seen to be $O(pqL^3)$. $\square$

4 Extensions

Channels arisen in building-layout design typically have the form shown in Figure 3. A channel is said to have *exits*, if it contains some net with outside connections; it is said to have *irregular boundaries*, if the blocks that form its sides are not horizontally aligned; and it is said to have *independent terminals*, if not all of its terminals are located on the boundaries of the blocks. In this section, we discuss briefly how our algorithm can be adapted to handle channels with exits, irregular boundaries and independent terminals. We will also show how to extend our algorithm to incorporate additional pin/block assignment constraints, which are often imposed by design rules and the design of previous phases.

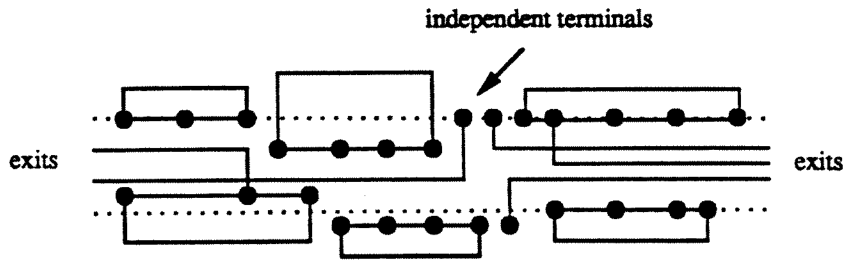


Figure 3: A typical channel arisen in building-block layout design

4.1 More General Forms of Channels

For each net with connection to the left of the channel, we introduce a pair of artificial terminals for this net, one on each side of the channel, and place it to the left of all other terminals on the same side of the channel. We can similarly introduce pairs of artificial terminals for nets with connections to the right of the channel. The introduction of these artificial terminals help to ensure that the contributions of the nets with outside connections to the local densities of the channel are appropriately accounted for. The length of the channel is also increased to an amount equal to the number of pairs of artificial terminals added. Note that a pair of artificial terminals is introduced for each outside connection because our algorithm cannot guarantee to place all the artificial terminals on one side of the channel to the left (right) of all terminals on the other side of the channel. Hence one artificial terminal per outside connection is not enough to guarantee the correctness of the algorithm because the local densities of the channel at some columns may not be correctly computed.

To handle channels with irregular boundaries, we choose a reference line for each side of the channel (the dotted lines in Figure 3) and change the definition of the local density of a channel at a column by adding or subtracting the irregularity of each side of the channel at that column with respect to the reference lines, depending on whether the boundaries of the channel at that column is above or below the reference lines. Since the boundaries of our channel is formed by the boundaries of the blocks, and the blocks are allowed to be shifted, the channel irregularity at a column depends on which block(s) the column intersects. In our algorithm we keep track of the corner terminals of the current block, so we can always determine the block(s) a column intersects and hence determine the channel irregularity at that column.

Independent terminals present no problem for our algorithm at all because terminals that are outside of the blocks can simply be treated as special blocks having length 1 and containing one single terminal (which is both the left and right terminal of the special block).

4.2 Additional Pin/Block Assignment Constraints

In many cases it is necessary to place additional constraints on where the terminals and blocks can be assigned. Note that block assignment constraints can be translated into pin assignment constraints on their corner terminals, hence we will only consider pin assignment constraints. Two kinds of pin assignment constraints introduced in [3, 5] namely, position constraints and separation constraints have been identified as being especially useful in modeling many situations. Position constraints are useful in modeling the cases where some terminals are forbidden to be placed in certain positions. Separation constraints are useful in modeling the cases where certain constraints exist between consecutive terminals (for example, the extremal terminals of consecutive blocks have to be at least certain distance away from each other in order to guarantee that the two blocks not assigned overlapping positions).

As in [3, 5], position constraints can be represented by associating with each terminal the subset of positions that it can be assigned to. Hence violation of position constraints can be easily checked. Therefore, to handle position constraints, all we need to do is to modify the definitions of the generalized crossing numbers so that they are set to $+\infty$ if position constraints are violated in that particular form of channel. This will not increase the worst case time complexity of the algorithm. Position constraints within the blocks can also be handled because positions inside a block can be determined by the distance from the left corner terminals of the block.

Separation constraints between consecutive terminals can be represented by associating with each pair of consecutive terminals a lower and a upper bound on the distance between them. General separation constraints are more difficult to handle because when consider a current terminal, we need to know where the terminal that is immediately to its left is assigned to. This forces us to introduce additional arguments to the generalized crossing numbers and the density function to remember the positions of the last assigned terminals.

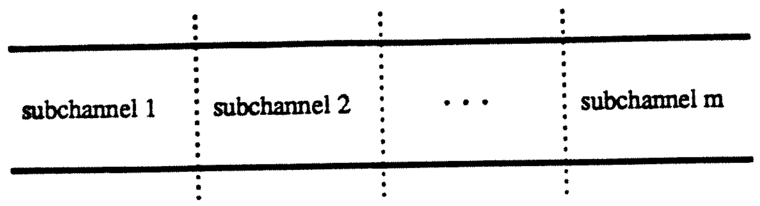
It can be achieved by replacing the argument k by two new arguments x and y which is the position of the last terminal on the top, bottom of the channel, respectively. With the help of these new arguments, violation of separation constraints can also be easily checked in a way similar to that described in the last paper. The complexity of the algorithm now becomes $O(pqL^5)$. In practice, it is the case the only special forms of separation constraints are needed to model many situations. For the special forms of separation constraints considered in [3, 5], our algorithm still works in $O(pqL^3)$ time.

5 Fast Heuristic Approaches

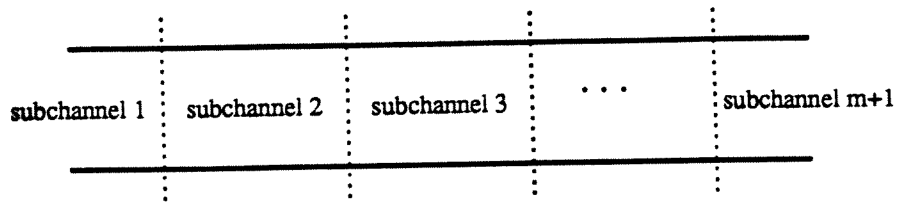
We have presented an optimal algorithm for minimizing channel density by shifting blocks and terminals and shown its extensions to practical situations. However, optimality is not absolutely necessary in many case if we can obtain results that are close to optimal. In these cases our algorithm can be used to develop fast heuristic algorithms. We describe several such heuristic approaches in this section. Our experimental results indicate that they are both efficient and effective.

First we tentatively fix the blocks and the terminals on one side of the channel, say the bottom of the channel, and then apply our algorithm to shift the blocks and terminals on the top of the channel. In doing so we can drop the second and fifth arguments from the function f and the generalized crossing numbers (because they are only related to the bottom of the channel), and therefore the complexity of the algorithm is reduced to $O(pL^2)$. We then apply our algorithm again, this time having the blocks and terminals on the top of the channel fixed to the positions obtained as above, and shift the blocks and terminal on the bottom of the channel. This takes $O(qL^2)$ time. This process can be repeated a constant number of times or until no further reduction in channel density is obtained, each time changing the role of the two sides of the channel. The complexity of this heuristic approach is $O((p+q)L^2)$. Note that this is about two orders of magnitude lower than the complexity of the optimal algorithm.

In the case of very long channels, we can reduce the complexity of the algorithm by partitioning the channel into m subchannels of roughly equal length and distributing the blocks evenly among the subchannels. We then apply our algorithm to each subchannel independently. Having shifted the blocks and terminals inside each subchannel, we can



(a) The original partition



(b) The new partition obtained by "shifting"

Figure 4: Partitions of a long channel

resume this process on a new partition of the channel obtained from the original one by horizontally shifting the partition to the right by an amount of one half of the length of a subchannel, as illustrated in Figure 4, so that in the next run blocks and terminals are allowed to be shifted across one subchannel of the original partition to another. Again this process can be repeated a constant number of times each time on a new partition obtained from the previous one by shifting to the right half the length of a subchannel, or until no further reduction in channel density is obtained. The complexity of this approach is

$$O(m * (p/m) * (q/m) * (L/m)^3) = O(pqL^3/m^4).$$

To further reduce the running time of the algorithm, we can combine the two heuristics above and thus reduce the complexity of the algorithm to

$$O((p+q)/m * (L/m)^2) = O((p+q)L^2/m^2).$$

The initial placement and distribution of blocks and terminals from which we start our heuristic algorithms can affect the optimality of the final results. To improve the quality of the solutions, we may apply our algorithm to several different initial solutions and select the best result we obtain.

6 Applications

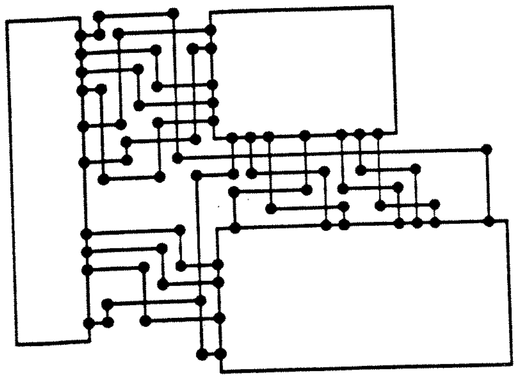
In the detailed routing phase of building-block layout design, the blocks and terminals typically can still be shifted without destroying the placement, the global routing or violating any design rules. Figure 5 shows that this kind of freedom can be used to obtain further area reductions. Figure 5(a) shows the layout without allowing the blocks and terminals to be shifted (area = 540). Figure 5(b) shows the layout obtained by allowing only the blocks to be shifted (area = 437), and Figure 5(c) shows the layout obtained by allowing both the blocks and terminals to be shifted (area = 396).

Our algorithm is particularly well suited for applications to detailed routing of building-block layout design where the placements are slicing structures [9]. Detailed routing is done one channel at a time. Our algorithm can be applied to compute positions for the blocks and terminals on the sides of the channel so that the density of the resulting channel is minimized. A detailed router is then used to complete the routing. When a channel is routed, an *intermediate building block* is formed which consists of the channel together with the blocks that form the sides of the channel. These intermediate building blocks are treated the same as basic blocks in later stages.

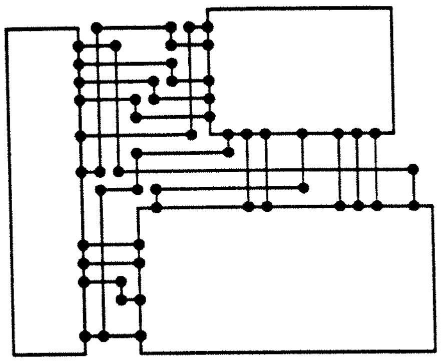
Although our algorithm is optimal with respect to a single channel, minimizing density of each individual channel may not lead us to a final layout with minimum area. In view of this, we generate several different implementations for each intermediate building block and store them for later use. Different implementations of intermediate building blocks can be generated by specifying different channel lengths in applying our algorithm. There is a trade-off between channel length and channel density. Long channels tend to have small densities because the blocks and terminals have more freedom to move. The different implementations of intermediate building blocks are then used in forming different implementations of higher level intermediate building blocks. Among the set of implementations generated for the entire layout, we choose the one with minimum area as our final result.

7 Experimental Results

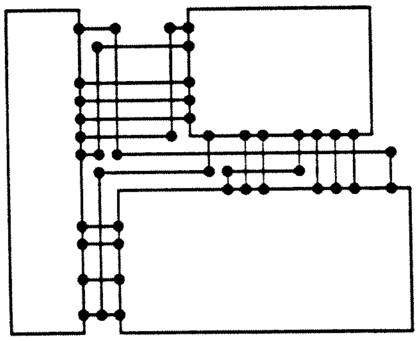
We have implemented our algorithm and the first heuristic approach in C language on a Sun SPARC station 1. Preliminary experimental results on channels are shown in Table 3.1.



(a) The original layout



(b) After moving blocks



(c) After moving blocks and terminals

Figure 5: A building-block layout example

Ex.	Length	#Terminals	Original	Optimal		Heuristic	
	L	$p+q$	density	density	time	density	time
Ex1	10	14	3	2	8.9	2	0.05
Ex2	29	30	6	2	306.5	3	1.1
Ex3	10	12	4	1	6.5	1	0.02
Ex4	15	20	5	1	26.4	1	0.1
Ex5	29	30	6	2	287.9	2	0.7
Ex6	34	40	6	4	1059.9	4	1.1

Table 1: Experimental results of Algorithm Minimizing_Channel_Density

Ex.	#Blocks	#Terminals	Old Area	New Area	Time	Reduction
Ex1	3	33	540	396	0.7	26.7%
Ex2	14	132	1512	1218	10.8	19.4%
Ex3	5	60	736	559	3.2	24.0%
Ex4	6	18	377	290	1.0	23.1%
Ex5	6	20	442	374	1.5	15.4%

Table 2: Experimental results on building-block layouts

CPU times were measured in seconds. As can be seen from the table, significant reductions in channel densities were obtained by both the optimal algorithm and the heuristic algorithm. The heuristic approach was able to obtain optimal or near optimal results most of the time while drastically reducing the running time.

Table 3.2 shows some results of applying our algorithm to detailed routing in building-block layout designs. These results were obtained by using the first heuristic algorithm instead of the optimal one.

8 Concluding Remarks

We present in this paper an optimal algorithm for a channel pin assignment problem where the blocks and the terminals of the channel are allowed to shift and the objective is to minimize channel density. Our study is motivated by many practical situations arisen in the layout design of VLSI circuits. Our algorithm can be easily adapted to handle situations that are often encountered in standard-cell and building-block layout designs. Based on our optimal algorithm we also propose fast heuristic approaches which run much faster than the optimal algorithm and produce results that are very close to optimal. These heuristic algorithms are especially useful in situations where optimality is not necessary and computational resources are critical. Preliminary experimental results of our algorithms are very encouraging. Substantial reductions in channel densities were obtained in moderate computation times.

Our algorithms have been successfully incorporated into detailed for building-block layouts. In such applications minimizing the density of an individual channel is no longer enough to guarantee the optimality of the final result. Interactions between channels have to be taken into consideration. One novel application of our algorithm is to generate different implementations of intermediate building blocks by specifying different channel lengths, so that when routing a next higher level channel, we have the freedom of choosing the best implementation of the blocks generated earlier.

Future research will focus on optimizing the current implementation of the algorithm, on wire length minimization, and on the generalization of our algorithm to the case of multiple parallel channels, as in standard cell layout design. Also of interest are extensions of our algorithms to more general forms of separation constraints, for example, separation constraints between nonconsecutive terminals and separation constraints between terminals on opposite sides of the channel. We would also like to consider cases where certain pin permutations are allowed, for example, if the blocks are allowed to be flipped.

References

[1] Y. Cai, *Efficient VLSI Layout Algorithms*, Ph.D. Dissertation, Department of Computer Sciences, University of Texas at Austin, May 1992.

[2] H. Cai and P. Dewilde, "Attacking the Problem of Minimizing Channel Density", *Proc. of the 1986 IEEE International Symposium of Circuits and Systems*, 353-356.

[3] Y. Cai and D.F. Wong, "An Optimal Channel Pin Assignment Algorithm", *Proc. of the 1990 IEEE International Conference on Computer-Aided Design*, 10-13.

[4] Y. Cai and D.F. Wong, "Minimizing Channel Density by Shifting Blocks and Terminals", *Proc. of the 1991 IEEE International Conference on Computer-Aided Design*, 524-527.

[5] Y. Cai and D.F. Wong, "Optimal Channel Pin Assignment", *IEEE Trans. on CAD, vol. CAD-***10**, 1413-1424, 1991.

[6] I.S. Gopal, D. Coppersmith and C.K. Wong, "Optimal Wiring of Movable Terminals", *IEEE Trans. on Comp., vol. C-***32**, 845-858, 1983.

[7] A.S. LaPaugh and R.Y. Pinter, "On Minimizing Channel Density by Lateral Shifting", *Proc. of the 1983 International Conference on Computer-Aided Design*, 121-122.

[8] B. Preas and M. Lorenzetti, ed., *Physical Design Automation of VLSI Systems*, The Benjamin/Cummings Publishing Company, Inc., Menlo Park, CA 1988.

[9] D. F. Wong and C. L. Liu, "A New Algorithm for Floorplan Design", *Proc. 23rd ACM-IEEE Design Automation Conference*, 101-107, 1986.