New iterative construction approach to routing with compacted area

C.-C. Tsai
S.-J. Chen
P.-Y. Hsiao
W.-S. Feng

Indexing terms: Placement, Routing, Compaction, Algorithms

Abstract: The new iterative construction approach presented in this paper consists of three algorithms, namely, channel expansion routing, track assignment, and module location refinement. These algorithms, contrary to the conventional methods implemented with a sequence of tools separately, are linked with a common data structure which guarantees a maximal compatibility. With an initial nonoverlapping placement as input, the iterative construction approach generates a final layout with more compacted area than the layout result from the one-dimensional compactor or some of two-dimensional compactors. Several layout examples in the literature are tested to show the effectiveness of our approach.

1 Introduction

In the literature, the most familiar methodology for solving a module (i.e. a hierarchical module or a building block) layout problem is formed with three sequential and distinct tools: the placement of modules [1–2], the interconnection of wires between modules [3–7], and the one- or two-dimensional compaction of the entire module [8–13]. A successful routing with minimal layout area depends on the locations of modules, the positions of the terminals on a module, and the space required for interconnections between modules. The space required for wiring in the intermodule area, however, cannot be determined until the routing design is complete (which, in turn, depends on the module placement). In general, a loosely placed environment is preferred to complete the routing, thus, it usually requires an extra compaction tool to obtain an improved output. Since these tools are handled separately, the problems of interface and structure compatibility between these tools (placement, routing and compaction) may be serious.

Some of the interface problems between placement and routing have been considered. Fukui [14] presented an estimated channels capacity as the basis for linking routing requirements with module placement, but this cannot guarantee a 100% routability. Ciesielski [15–16] proposed a loosed combination of a relaxed digraph and a two-dimensional router, and attempted to obtain the two-dimensional placement of IC modules which minimises the layout area, but the interface compatibility problem between the module placement and the two-dimensional router still exists. Dai [17–18], Li [19], and Xiong [20] proposed a method that can reduce the space area by refining the original placement based on a mixed-adjacent graph model, however, the modified placement may reduce the superiority of the original global routing and increase the complexity of detail routing [21–23].

In this paper, a new iterative construction approach is proposed to solve the problems mentioned above. This iterative method is mainly composed of three algorithms, i.e. channel expansion routing, track assignment, and module location refinement. These algorithms, instead of being implemented as a sequence of tools separately, are built on a common data structure (H–V model of corner-stitching) which promises a maximal compatibility.

Fig. 1 shows an overview of the method. The input is a nonoverlapping (loose or congested) placement result. The channel expansion routing is first involved. This is the global routing stage which tries to route logically all

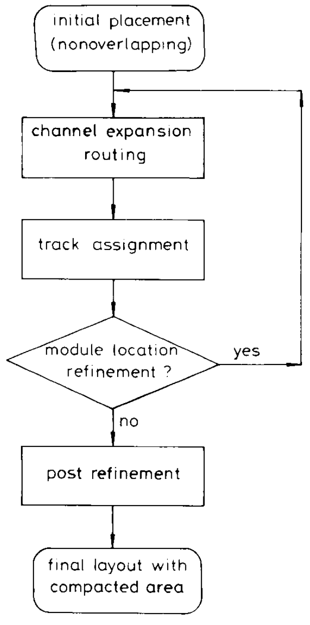

initial placement (nonoverlapping)
channel expansion routing
track assignment
module location refinement ?
yes
no
post refinement
final layout with compacted area

Fig. 1 *Overview of the new iterative construction approach*

Paper 7717E (C2), first received 8th November 1989 and in revised form 5th July 1990

C.-C. Tsai is with the Department of Electronic Engineering, National Taipei Institute of Technology, Taipei, Taiwan, Republic of China

S.-J. Chen and W.-S. Feng are with the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, Republic of China

P.-Y. Hsiao is with the Department of Computer & Information Science, National Chiao Tung University, Hsinchu, Taiwan, Republic of China

the nets without violating the channel capacity constraint. Secondly, the assignment of tracks to the horizontal and vertical channels is performed. Thirdly, the module location refinement is followed, which uses a diagonal relaxation digraph for the estimation of module movement and the determination of module locations. The iteration continues until all modules keep their location fixed and all nets are successfully routed (that is, the routing space cannot be reduced again). Finally, the post refinement is performed to correct any design-rule violated routing segments. A final layout with compacted routing area can thus be obtained.

The new iterative construction approach guarantees a final layout with more compacted area than any layout result from the conventional solutions (even those with a one-dimensional compactor). Several layout examples in the literature are tested; these experimental results showed that our approach is better than some of the two-dimensional compactors which are complicated and time consuming.

2 Common data structure

The success of our iterative construction approach depends completely on a sequence of interacted algorithms, namely channel expansion routing, track assignment, and module location refinement. To maximise the compatibility between these algorithms, the choice of a common data structure will play an important role.

Corner-stitching [24] is our best choice, because it is a very powerful data structure for the finding of a nearest neighbour and it supports many fast algorithms, such as insertion, deletion, neighbour searching, region searching, visibility searching, and channel finding.

Based on the slicing method of the corner-stitching, we developed an H–V model (horizontal and vertical maximum strips) of corner-stitching as the common data structure for these three algorithms, which can support both the horizontal and vertical space tile representations simultaneously. Figs. 2*a* and *b* illustrate examples for the H-plane and V-plane of the H–V model, respectively. The H–V model can explicitly represent the empty spaces and link the tiles of various types (e.g. space tile and solid tile) together at their corners like a patchwork quilt. For instance, in Fig. 2*a* and *b*, there are four pointers, *rt*, *tr*, *lb*, *bl*, in the right-top and left-bottom corners of the solid tile *B1*. An H-plane (V-plane) is composed of modules, H-wire (V-wire) segments, and contacts (crosses between horizontal and vertical wires). A solid tile is either a module, an H-wire, a V-wire, or a contact. Other empty areas are called routing areas which consist of H-space and V-space tiles in the H–V plane. The space tiles were also known as spacing channels or channels. The channel in the H-plane is called H-channel, whereas in the V-plane it is called V-channel.

With the H–V model of corner-stitching, the searching task of the routing stage would become simpler and more efficient, because the H–V model can significantly reduce the number of channel searching by alternating the manipulations on the H-plane and V-plane. Also, from the fact that the H–V model inherently supports the representation of H-channels and V-channels, the tasks of channel expansion routing and track assignment operated on these channels can be implemented effectively. Furthermore, the H–V model also supports the module visibility searching capability (two solid tiles are said to be mutually visible if one H-channel or V-channel exists between them and they are not blocked by any other solid tiles). These characteristics of module visibility searching will help in setting up the diagonal relaxation digraph and to estimate the new location for each module. Therefore, using the H–V model provides an excellent environment which promises a maximal compatibility among the three algorithms of the iterative construction approach.

In addition, to simplify the implementation of these three algorithms, we assume that the main-module (or the chip) is located on a virtual grid which has the following characteristics:

(i) The spacing between grid lines W_g is the sum of a minimum wire width and a minimum distance between two parallel wires.

(ii) All the component locations are represented on the virtual grid. Here, the components include the (hierarchical or functional) modules, H-wires, V-wires, and contacts (or vias).

(iii) The boundaries and the terminals of each (hierarchical) module are assumed to be on the grid lines and the grid points, respectively.

(iv) Two wires on the H–V plane can be connected by a contact (via) through a grid point.

(v) Power/ground wires are represented by wide wires, which occupy more than one grid width or height.

Although the virtual grid layout model is used, little extra memory spaces are required because all the major operations related to the three algorithms work on the tiles in the H–V plane but not on the grid lines, and the grid model is only used for the estimation of tracks position and modules location.

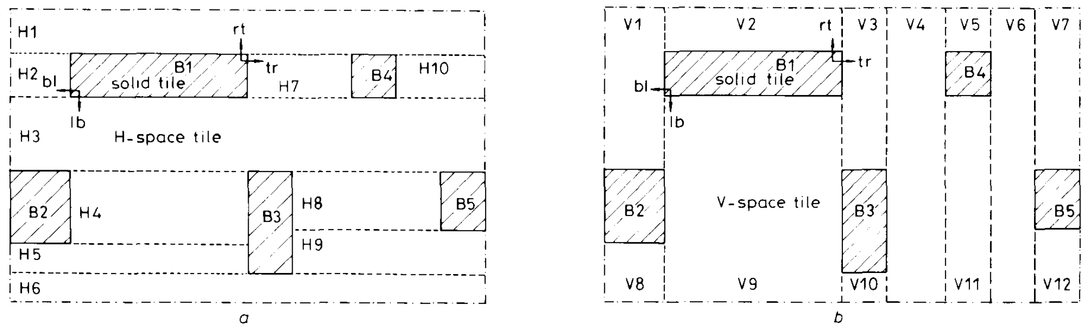


Fig. 2 *Improved H–V model of corner stitching*
a H-plane with maximal horizontal strips
b V-plane with maximal vertical strips

3 Channel expansion routing

The channel expansion routing (CER) [7] is a global router that logically wires one net at a time and tries to get a net path with a minimum of both the Manhattan distance and the number of bends. It attempts to find the shortest net path between any two terminals or a near-optimum steiner tree among multiple terminals. A net path is formed by a sequence of segments passing through channels with large enough track capacity. Since the task of the CER algorithm is to attempt to find a reasonable assignment of segments on tracks through a channel, it must satisfy the constraint of the channel capacity. Any net routing is said to have failed if any piece of its component segments cannot satisfy the channel capacity constraint. Naturally, when the placement of modules is too crowded, the module location needs refining.

The main components of the CER algorithm are its H–V channel expansion and its cost function as described in Section 3.1.

3.1 H–V channel expansion

Based on the H–V model, a Manhattan path between two (source and target) terminals is composed of a sequence of alternating H (V) and V (H) channels in the H–V plane. An H–V channel expansion can thus be used to find the Manhattan path by alternately performing the H (V) and V (H) channel expansion. A point which guides an expanding channel to expand toward this point itself is called a guiding point *Pg*. The guiding point must be located on the cross road of an H and a V channel which are called guiding channels *Hg* and *Vg*, respectively. Before the H–V channel expansion, the neighbouring channel of the source terminal forms an initial H or V channel, and the guiding point and guiding channels are determined by the neighbouring channel of the target terminal in the H–V plane. During the H–V channel expansion, an expanding tree is constructed. Beginning from an initial channel (the root node of the expanding tree), some of the channels are chosen to expand toward the guiding channels. The expanding work cannot be terminated until any one of the guiding channels (*Hg* in the H-plane or *Vg* in the V-plane) is reached. Finally, the Manhattan path can be obtained by backtracking from the guiding channel to the initial channel in the expanding tree.

Fig. 3 shows an example of the H–V channel expansion, which routes a two-terminal net from *B1.1* to *B3.1*. In Fig. 3*a*, the H-channel *H4* in the H-plane, and the V-channel *V9* in the V-plane, are the guiding channels (i.e. *Vg* and *Hg*) which can be found by the guiding point *Pg* located at the left side of the target terminal *B3.1*. The H-channel *H2* in the H-plane is an initial channel which can be found by the left side of the terminal *B1.1*. The H–V channel expands from the initial channel *H2* toward the guiding channels *H*4 and *V9*. Fig. 3*b* shows the expanding tree with the rooted node *H*2; the channels with thin broken lines are not used during the H–V channel expansion, and the heavy solid lines denote the connection path of the net, i.e. $H2 \rightarrow V1 \rightarrow H3 \rightarrow V9 \rightarrow H4$. This connection path is the shortest one connecting the two terminals *B1.1* and *B3.1*, as shown in Fig. 3*a*.

To obtain a proper selection during the H–V channel expansion, the modified A* technique [25] and damping concept are investigated. The heuristic cost function *f* is defined as

$$f = g + h + \rho \tag{1}$$

where *g* is the connection length from the initial channel to the current expanding channel, *h* is the connection length from the current expanding channel to the guiding point, and ρ is the damping length from the current expanding channel to the guiding point, defined below:

$$\rho = \sum_{i=1}^{m} (\eta \cdot W_d + W_{Bi}/2) \tag{2}$$

where *m* is the total number of dampers (in general, a solid tile is considered as a damper) that lies on the rectilinear connection path from the current expanding channel to the guiding point, W_d is the width (height) of a wire, W_{Bi} is the width (height) of a damper *Bi*, and η is the damping factor (a positive real number between 1.0–3.0, a large η represents selecting a path with more total length and less bends, whereas a smaller η is for selecting a path with less total length and more bends).

The following calculations (eqns. 3 and 4) depict the finding of cost functions f_{H1} and f_{H3} of the H-channel *H1* and *H3* in Fig. 3*b*, respectively.

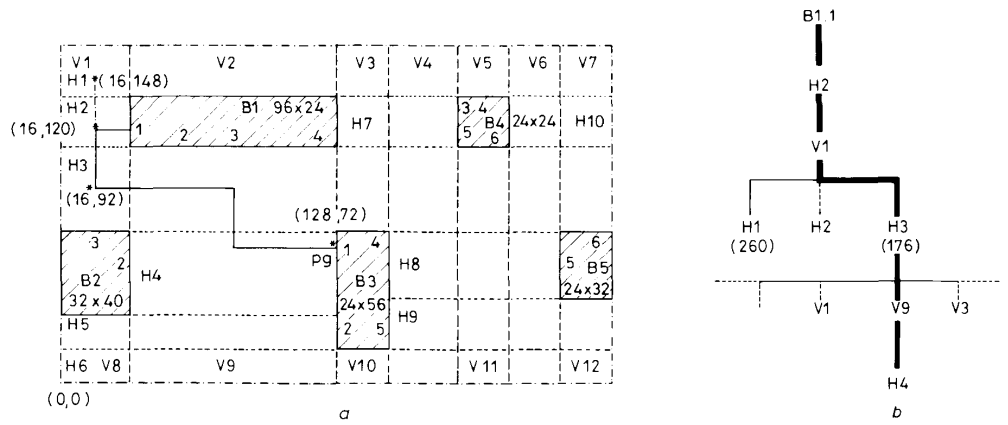


Fig. 3 *H–V channel expansion for routing a two-terminal net connecting B1.1 and B3.1*
a Connection path
b Expanding tree

$$g_{H1} = g_{V1} + |H1[y] - H2[y]|$$
$$= 16 + |148 - 120| = 44$$
$$h_{H1} = |Pg[x] - H1[x]| + |Pg[y] - H1[y]|$$
$$= |128 - 16| + |72 - 148| = 112 + 76 = 188 \quad (3)$$
$$\rho_{H1} = \eta \cdot W_d + B3[width]/2 = 2.0 \cdot 8 + 24/2 = 28$$
$$f_{H1} = g_{H1} + h_{H1} + \rho_{H1} = 44 + 188 + 28 = 260$$

and

$$g_{H3} = g_{V1} + |H3[y] - H2[y]|$$
$$= 16 + |92 - 120| = 44$$
$$h_{H3} = |Pg[x] - H3[x]| + |Pg[y] - H3[y]|$$
$$= |128 - 16| + |72 - 92| = 112 + 20 = 132 \quad (4)$$
$$\rho_{H3} = 0$$
$$f_{H3} = g_{H3} + h_{H3} + \rho_{H3} = 44 + 132 + 0 = 176$$

Moreover, while a segment passes through a channel or turns to the orthogonal direction during the H–V channel expansion, it must satisfy the constraint of channel capacity. Therefore, at the intersected edges (called intervals) between the current channel and some of its orthogonal channels, their capacity constraints have to be checked. For example, in Fig. 4, the horizontal segment hw_1 will pass through the H-channel $H1$ and turn to the V-channels; there are three intervals, $I_1(H1, V1)$, $I_2(H1, V2)$, and $I_3(H1, V3)$, in which capacity constraints have to be checked.

3.2 Algorithm

The H–V channel expansion, discussed previously, is only appropriate for routing a two-terminal net. However, it can also be extended for the multiple-terminal net. First, determine the guiding channels (Hg in the H-plane and Vg in the V-plane) from the guiding point Pg closest to the centre of the multiple terminals of the net. Then, execute the H–V channel expansions from every terminal to the guiding channels, set up a net-forest structure consisting of a number of expanding trees, and record the incident points in the guiding channel Hg or Vg, if any exists. These incident points are used to determine the future orthogonal connections through the guiding point [7]. Finally, find the connection path from the net-forest structure.

Since the H–V channel expansion always selects the channel with a minimal cost and satisfying the channel capacity as the next expansion channel, it attempts to find the shortest path between the two terminals. Sometimes, if it cannot find a channel to be expanded in the current expansion, the H–V channel expansion tries to choose its sibling channels or to perform a backward expansion from the current channel until the new current channel is found, if it exists. Otherwise, this net is declared a failed net and pushed into a failed-net stack. The failed nets in the stack will be picked and rerouted without the consideration of the channel capacity later on.

Now, the CER algorithm can be described as follows:

```
**CER_Algorithm ( )**
Let a $STACK$ be empty;
FOR $i = 1$ TO $q$   /* $q$ is the number of nets */
{  $ROUTE$ = TRUE;
   IF (Term_no($net_i$) = 2)   /* a two-terminal net */
   Then
   {  Get the source and target terminals $S\_term$ and $T\_term$,
      respectively, from the $net_i$;
      HV_channel_expansion ($S\_term$, $T\_term$);
      If ($ROUTE \neq$ TRUE)
      THEN   Push the $net_i$ to a $STACK$;
      ELSE   Find the connected path from the expanding
             tree of the $net_i$;
   }  ELSE   /* the multiple-terminal net */
   {  Determine the guiding channels ($Hg$ in the H-plane and
      $Vg$ in the V-plane) from the guiding point $Pg$ closest to
      the center of the multiple terminals of the $net_i$;
      FOR $j = 1$ TO $p$   /* $p$ is the number of terminals
                              in the $net_i$ */
      {  **HV_channel_expansion** ($term_j$, $Pg$);
         Extra incident points are recorded in the guiding
         channel $Hg$ and $Vg$ during the H–V channel expansion;
      }
      IF ($ROUTE \neq$ TRUE)
      THEN   Push the $net_i$ to a $STACK$;
      ELSE
      {  Find the connected path from the net-forest
         structure of the $net_i$, and add some pieces of wire
         segments that depend on these incident points if
         any one exists;
      }
   }
}
IF (The $STACK$ is not empty)
THEN   Reroute each net in the $STACK$ without considering
       channel capacity;
```

```
HV_channel_expansion (Source, Target)
    Alternatively perform the H and V channel expansions from the
    source to the target terminal; (see Section 3.1)
    IF (the expansion failed)
    THEN  ROUT = FALSE;
}
```

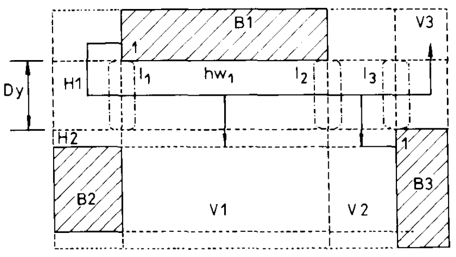


Fig. 4 *Three intervals, I_1, I_2, and I_3, are checked when horizontal segment hw_1 is passing the H-channel H1 and turning to the orthogonal directions.*

The time complexity of the CER algorithm, described as above, is $O(qN)$, where q is the number of input nets and N is the number of solid tiles in the H–V plane.

Fig. 5 shows an example of the channel expansion routing. The net 5 consists of three terminals with terminal ordering such as *B3.5*, *B4.5*, and *B5.5*. The guiding channels are *H8* in the H-plane and *V11* in the V-plane and the guiding point *Pg* is just the point located at the centre of channels *H8* and *V11*, as shown in Fig. 5*a*. Fig. 5*b* shows the three expanding trees constructed from every terminal to the guiding point *Pg* by using the H–V channel expansion. The net 5 path is obtained by the combination of these three expanding trees. In addition, there is one incident point *P1* in the H-guiding channel *H8*, which determines the two orthogonal connections through the guiding point *Pg*. The final connection path of the net 5 is shown in Fig. 5*a*.

Fig. 5*a* also shows the corresponding global layout on a 6-net forest. All of its net paths are shown in Fig. 5*c*, each link of solid lines indicates a net path, and links of dotted lines represent resident nets in a channel. For example, the net 5 connects 3 terminals (*B3.5*, *B4.5*, and

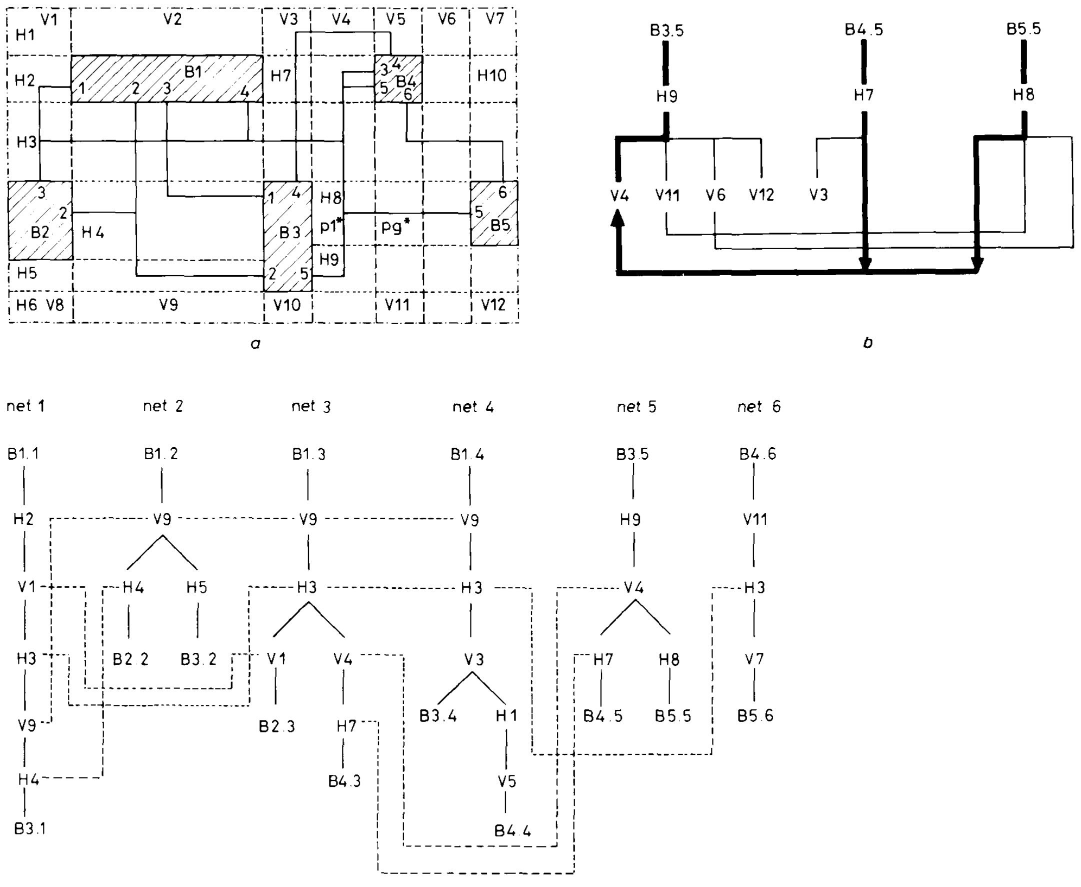


Fig. 5 *Example of the channel expansion routing*
a global layout
b net-forest structure consists of three expanding trees of the net5, which connects three terminals
c net paths

net 1 = {B1.1, B3.1}	net 4 = {B1.4, B3.4, B4.4}
net 2 = {B1.2, B2.2, B3.2}	net 5 = {B3.5, B4.5, B5.5}
net 3 = {B1.3, B2.3, B4.3}	net 6 = {B4.6, B5.6}

B5.5) with a sequence of channels *H9*, *V4*, *H7*, and *H8*, and four nets (net 1, net 3, net 4, and net 6) reside in the H-channel *H3*.

Note that in the CER algorithm, the order of constructing the expanding trees depends on the terminal order. The current expanding tree can use the channel again which was not a part of the net path at the previous expanding trees. Hence, a channel is stored in only one of the nodes among the expanding trees. This can significantly reduce the memory redundancy and protect the H–V channel expansion from the divergence of memory allocation. As stated in Reference 7, the CER algorithm owns the following properties:

(i) acyclic expansion
(ii) minimal cost decision
(iii) backward expansion channel selection
(iv) expanding tree combination.

In addition, three strategies as below are used to speed up this algorithm:

(i) constrained expansion area
(ii) limited expansion depth
(iii) oriented expansion direction

With these properties and strategies, the CER algorithm is capable of finding a near-optimal connection path in the net-forest structure.

4 Track assignment

To support the cost estimation at the stage of module location refinement, a track assignment (TA) (or channel assignment) stage needs to be investigated first. The goal of track assignment is to find the track distribution for each horizontal and vertical channel such that the required tracks are kept minimal.

4.1 Model

Generally speaking, the track assignment problem can be viewed, given a set of segments *n* (pairs of real number), as how to find a minimal partition of this set such that no elements of the partition occupy two overlapping intervals. A so-called left-edge algorithm (LEA) is a well known method for this problem. Also, Gupta [26] presented a $\theta(n \log n)$ algorithm which is optimal. Our track-assignment problem is based on this concept but with more practical consideration, such as the routing orientation of the two endpoints of a segment, the extra-track rule of a channel (introduced in the following Section), and the length (width) of a segment. This kind of information is very helpful to determine a proper track for each segment, to eliminate the number of crossings among segments, and to reduce the number of tracks in the adjacent channels.

When estimating the capacity or density of an H-channel (V-channel), the bottom (left) boundary of an H-channel (V-channel) that does not belong to the top (right) boundary of a module or the bottom (left) boundary of a main-module, is considered as an extra track at the track assignment stage. And two adjacent H-channels (V-channels) form an adjacent boundary which can be used to assign some H-segments (V-segments) crossing the two channels at the same time. To avoid the potential overlapping of segments at this boundary, we assume that the boundary is appropriate to serve as an extra track for the top H-channel (right V-channel), i.e. the boundary cannot be assigned to any H-segments (V-segments) for the bottom H-channel (left V-channel). This is called an extra-track rule of a channel. For example, in Fig. 4, the intervals $I_1(H1, V1)$ and $I_2(H1, V2)$ have an extra track, i.e. its density (no. of tracks) is equal to Dy/W_g, where Dy and W_g represent the height of an H-channel *H1* (note, Dx is used for the width of a V-channel) and the grid width (height), respectively. However, the interval $I_3(H1, V3)$ has no extra track, i.e. their density is equal to $(Dy/W_g - 1)$. As shown in Fig. 6, obtained from Fig. 4, the segment hw_1 can be assigned to the bottom boundary of the H-channel *H1* and in turn to the V-channel *V2* to avoid causing potential overlapping of segments between the H-channels *H1* and *H2*, then the H-channel *H2*.

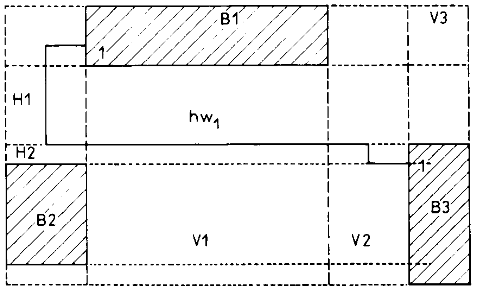


Fig. 6 *Result of track assignment obtained from Fig. 4*

With the above discussion, the model for the H-channel (V-channel) track assignment can be summarised in Fig. 7; the heavy solid lines in the figure cannot be assigned to any H-segments, but the dotted lines are open for all the H-segments. Of course, some segments will be influenced by fixed terminals at the boundary positions.

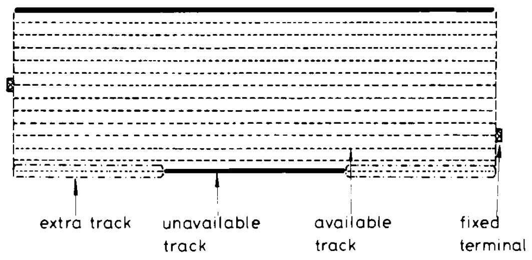


Fig. 7 *Model of an H-channel track assignment*

4.2 Algorithm

Both H-channel and V-channel track assignments exist in the H–V plane. Either one of them can be performed first; here, our approach adopts the H-channel track assignment first (according to experiments, the ordering of H-channel or V-channel track assignment first makes little difference toward a final result with more compacted area). With this ordering, if there are some rule-violated routing segments, the rip-up and rerouting techniques, such as jog or dog-leg segments inserting and wire segment moving, are used to solve them.

Also, one of the special cases at the stage of the H–V channel expansion, discussed in Section 3.1, is shown in Fig. 8*a* where it causes no problem for the channel capacity constraint, but there does exist the problem of potential overlapping of segments. Fortunately, these problems can be solved by inserting jog or dog-leg segments at this track assignment stage, and moving the related module at the stage of module location refinement (discussed in Section 5), as shown in Figs. 8*b* and *c*, respectively.

Without loss of generality, only the H-channel track assignment is discussed in detail here. With the model of the H-channel track assignment, the track assignment algorithm is described as follows:

```
TA_algorithm ( )
{
   FOR $i = 1$ TO $Hm$   /* $Hm$ is the number of the H-channels */
   {  H_track_assignment ($H$-$channel_i$);
   }
   FOR $j = 1$ TO $Vm$   /* $Vm$ is the number of the V-channels */
   {  Rotate the degree of 90 at the center of the $V$-$channel_j$;
      H_track_assignment ($V$-$channel_j$);
      Inversely rotate the degree of 90 at the centre of the
      $V$-$channel_j$;
   }
}
```

H_track_assignment (*H-channel*)

Initialisation: Assume that a set of n H-segments exist in the H-channel, and each H-segment belongs to a net and is located on the grid line, e.g. $\{(lx_i, rx_i, y_i) \mid, 1 \leqslant i \leqslant n\}$ represents the set of n H-segments each with a left endpoint (lx_i, y_i) and a right endpoint (rx_i, y_i) and $lx_i < rx_i$. Each endpoint of the H-segment has a routing orientation type. The left (right) type value, lv_i (rv_i) can be either *up*, *down*, *fix*, or *up-down* according to the position of its adjacent vertical wire. Let CT be the number of current track.

```
{  Sort the $lx$ of $n$ H-segments such that their order is
   $lx_i \leqslant lx_j$ for $i < j$;
   If the H-channel has an extra track, let $CT$ be $0$;
   otherwise let $CT$ be $1$.
   FOR $i = 1$ TO $n$
   {  If ($lv_i = fix$ OR $rv_i = fix$)
      THEN
      {  $YY = y_i/W_g$;
         IF ($YY < CT$)
         THEN
         {  IF (no orientation type among the H-segments
               of the $YY$th track is $fix$)
            THEN
            {  Update the $YY$th track of these H-segments
               to the number of $CT$; (See Fig. 9)
            }ELSE
            {  Determine the shape of jog or dog-leg
               segments depend on the orientation type
               of the related H-segments; (see Fig. 8)
            }
         }
         $ct_i = YY$;
      }  ELSE
      IF ($lv_i \neq fix$ AND $rv_i \neq fix$)
      THEN
      {  Let $rx_j$ be the rightmost endpoint among H-segments
         on the current track;
         IF ($lx_i = rx_j$)   THEN   $lx_i = rx_j + W_g$;
         $ct_i = CT$;
      }ELSE
      {  $CT = CT + 1$;
         $ct_i = CT$;
      }
   }
}
```

The time complexity of the TA algorithm is $O((Hm + Vm)n \log n)$, where Hm and Vm are the number of H-channels and V-channels, respectively.

4.3 Track cost

Note that this track-assignment algorithm, discussed previously, does not consider the reduction of the number of crossings between the horizontal and vertical segments.

To reduce the number of crossings, a track-assignment refinement is performed by attaching a track cost to each track. The cost ϕ_k of the kth track consists of two major parts defined as follows:

$$\phi_k = \beta + \gamma \tag{5}$$

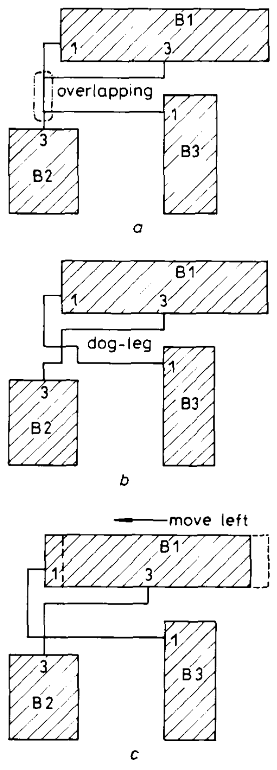


Fig. 8 *Overlapping case and its solutions*
a Overlapping case
b By inserting dog-leg segments
c By moving the module B1 to the left

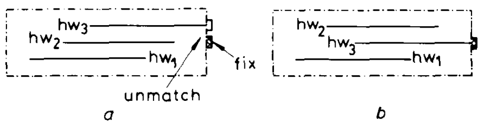


Fig. 9 *Horizontal track assignment for three horizontal segments*
a Error track assignment
b Final track assignment

β: The sum of endpoint orientation values of all segments that belong to the kth track, defined as:

$$\beta = \sum_{i=1}^{s} (lv_i + rv_i) \quad (6)$$

where s is the number of segments in the kth track; lv_i and rv_i are the orientation values of the left and right endpoint of each segment, respectively, and the orientation (type) values of the up (right), down (left), up-down (right-left), and fix endpoints of a horizontal (vertical) segment is $+0.5$ (let W_g be 8, four $1/W_g$ are chosen, i.e. $4(1/8) = 0.5$), -0.5, 0, and a large value of ψ, respectively. If one of the endpoints have the value ψ, β is equal to ψ. This means that all of the segments in the kth track cannot be changed.

γ is a real number of segment lengths normalised by the maximal track length in the channel, defined as follows:

$$\gamma = \frac{LS_k - \max\{LS_1, LS_2, \ldots, LS_k, \ldots\}}{\max\{LS_1, LS_2, \ldots, LS_k, \ldots\}} \quad (7)$$

where LS_k is the total length of the segments in the kth track.

With this track cost function, the H-channel (V-channel) track assignment will generate a better result for each channel. When the track-assignment refinement is performed on a H-channel (V-channel), it should not cause any overlapping segments on its own channel or its adjacent channels. Assume some segments have already been assigned to the kth track of a channel. If the calculated track cost ϕ_k of this kth track is the smallest (largest) one, these segments will be reassigned to the lowest (highest) track, but when ϕ_k is equal to ψ the number of segments in the kth track will keep their position. An example is shown in Fig. 10a, which is required for up to 12 crossings, its three track costs are found as

$$\begin{aligned}
&\text{Since } \max\{37W_g, (21+8)W_g, (7+8+8)W_g\} \\
&\quad = \max\{37W_g, 29W_g, 23W_g\} = 37W_g \\
&\phi_1 = (-0.5 - 0.5) + (37W_g - 37W_g)/37W_g = -1 \\
&\phi_2 = (-0.5 + 0.5 - 0.5 + 0.5) \\
&\qquad + (29W_g - 37W_g)/37W_g = -0.22 \\
&\phi_3 = (-0.5 - 0.5 - 0.5 - 0.5 - 0.5 + 0.5) \\
&\qquad + (23W_g - 37W_g)/37W_g = -2.38
\end{aligned} \quad (8)$$

With these track costs, a final result of the horizontal track assignment is obtained in Fig. 10b, which is only required for four crossings.

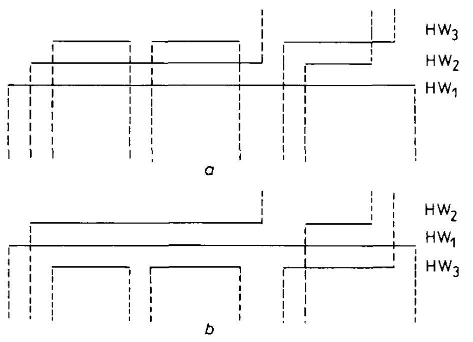


Fig. 10 *Example of track assignment*
a Original result of track assignment
b Result of track-assignment refinement

Another example, Fig. 11b shows the track assignment results for all the horizontal and the vertical channels in Fig. 11a.

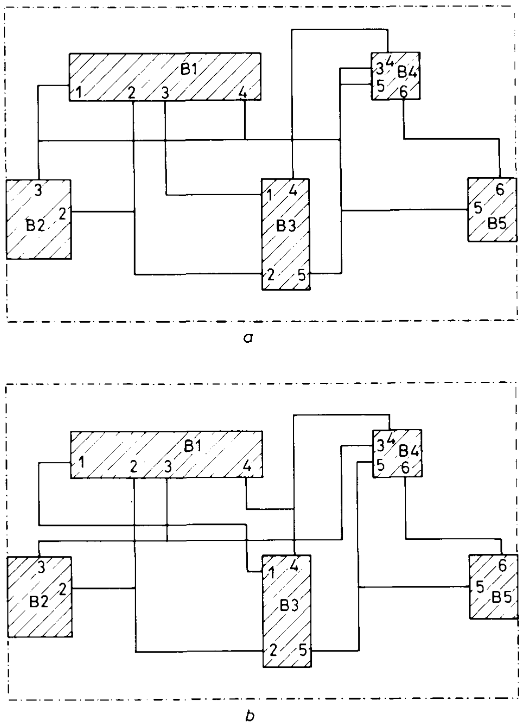


Fig. 11 *Further example of track assignment*
a Global layout result after the channel expansion routing
b Result after the track assignment

5 Module location refinement

An initial layout result has been obtained after performing both the channel expansion routing and the track assignment algorithms. However, it is possible that the layout result is either too loose or too crowded. We need a model, called diagonal relaxation digraph (DRDG), to estimate the layout result, and to refine the module to reasonable locations such that the final layout is correct and with more compacted routing area.

5.1 Diagonal relaxation digraph

Since a number of modules are visible from the left or the bottom side of a current module, the DRDG can be constructed by the horizontal (left) or the vertical (bottom) visibility searching of this module. For example, in Fig. 3a, module *B1* is visible from the left side of the module *B4* through the H-channel *H7*. In the DRDG, a module can have eight moving directions each of which depends on the locations of adjacent modules and on the resident tracks of neighbouring channels, as shown in Fig. 12. The displacements sx and sy of the module B_k are defined as

$$\begin{aligned}
sx &= B_j[x2] + rx - B_k[x1], \quad rx = W_g(tx + \tau) \\
sy &= B_i[y2] + ry - B_k[y1], \quad ry = W_g(ty + \tau)
\end{aligned} \quad (9)$$

where rx is the width (length) of the resident tracks tx between B_j and B_k and ry is the height (length) of the resident tracks ty between B_i and B_k. Whether τ is either 1 or 0 depends on the tx (or ty) being the number of resident tracks between two modules or between a module and the boundary of the chip, and [$x1$], [$x2$], [$y1$], and [$y2$] represent the boundary positions of the left, the right, the bottom, and the top side of a module,

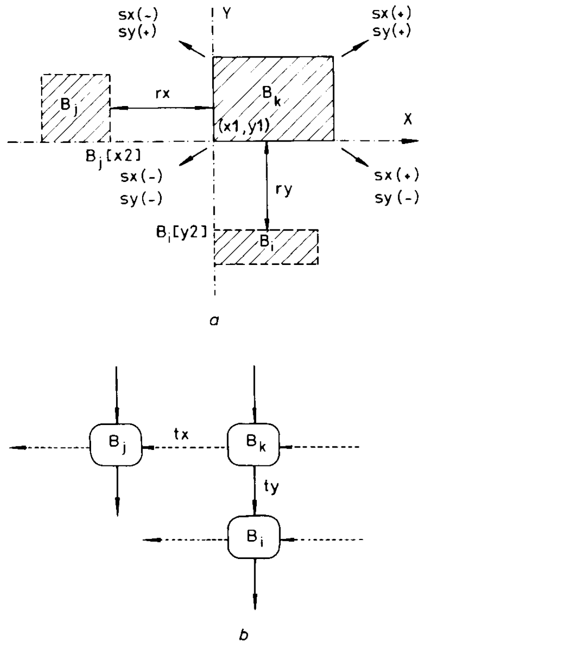


Fig. 12 *Block B_k and its digraph*
a Block B_k with either one of eight movements that depends on the positions of blocks B_i and B_j
b Diagonal relaxation digraph

respectively. Whether the module B_k has to be moved to the right (i.e. X-expand) or to the left (i.e. X-compact) depends on the sign of sx. Similarly, the module B_ks moving to the top (i.e. Y-expand) or to the bottom (i.e. Y-compact) depends on the sign of sy. Hence, each module might have eight moving directions (i.e. $X - Y$ or two-dimensional expand or compact), each of which depends on the combination of sx and sy. Since there are many modules, such as the number m of B_j and the number n of B_i, that are visible at the same time from the left and the bottom side of a module, the general formula of displacements sx and sy of the module B_k are redefined as

$$\begin{aligned} sx &= \max\{(B_{j1}[x2] + rx_{j2}), \\ &\qquad \ldots, (B_{jm}[x2] + rx_{jm})\} - B_k[x1] \\ sy &= \max\{(B_{i1}[y2] + ry_{i2}), \\ &\qquad \ldots, (B_{in}[y2] + ry_{in})\} - B_k[y1] \end{aligned} \tag{10}$$

Since the H–V model of corner-stitching inherently supports both horizontal and vertical module visibility searchings simultaneously, a DRDG for all the modules can be easily constructed from this model. The DRDG $G = (V, E)$ consists of a set of modules $V = \{v_1, v_2, \ldots\}$, called module vertices, and another set of $E = \{e_1, e_2, \ldots\}$, which elements are horizontal or vertical edges. There are four dummy vertices v_E, v_W, v_S and v_N located in the east, west, south, and north of the DRDG, respectively. These dummy vertices are viewed as the boundaries of the chip size. The number attached on each horizontal and vertical edge represents the number of track requirements between both adjacent vertices from top to bottom and from right to left, respectively. For instance, Figs 13*a*, *b*, and *c* depict the horizontal directed graph, the vertical directed graph, and the DRDG of Fig. 11*b*, respectively.

5.2 Algorithm

The module location refinement algorithm is operated on the DRDG which corresponds to the topological placement of the modules and is used to determine the new locations of modules and to modify the topological module relations. Whether a routing channel is compacted or relaxed depends on the loose or the crowded placement of modules, respectively. In the DRDG, some competitive areas (CAs) located at the cross places between the horizontal and vertical edges may exist, each of which is shared by a number of moving modules and thus causes these modules to overlap explicitly. However,

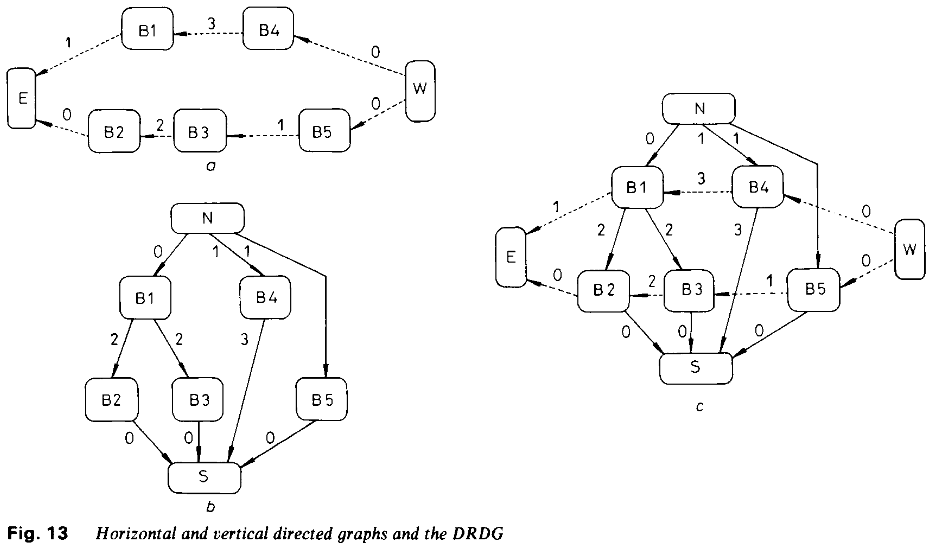


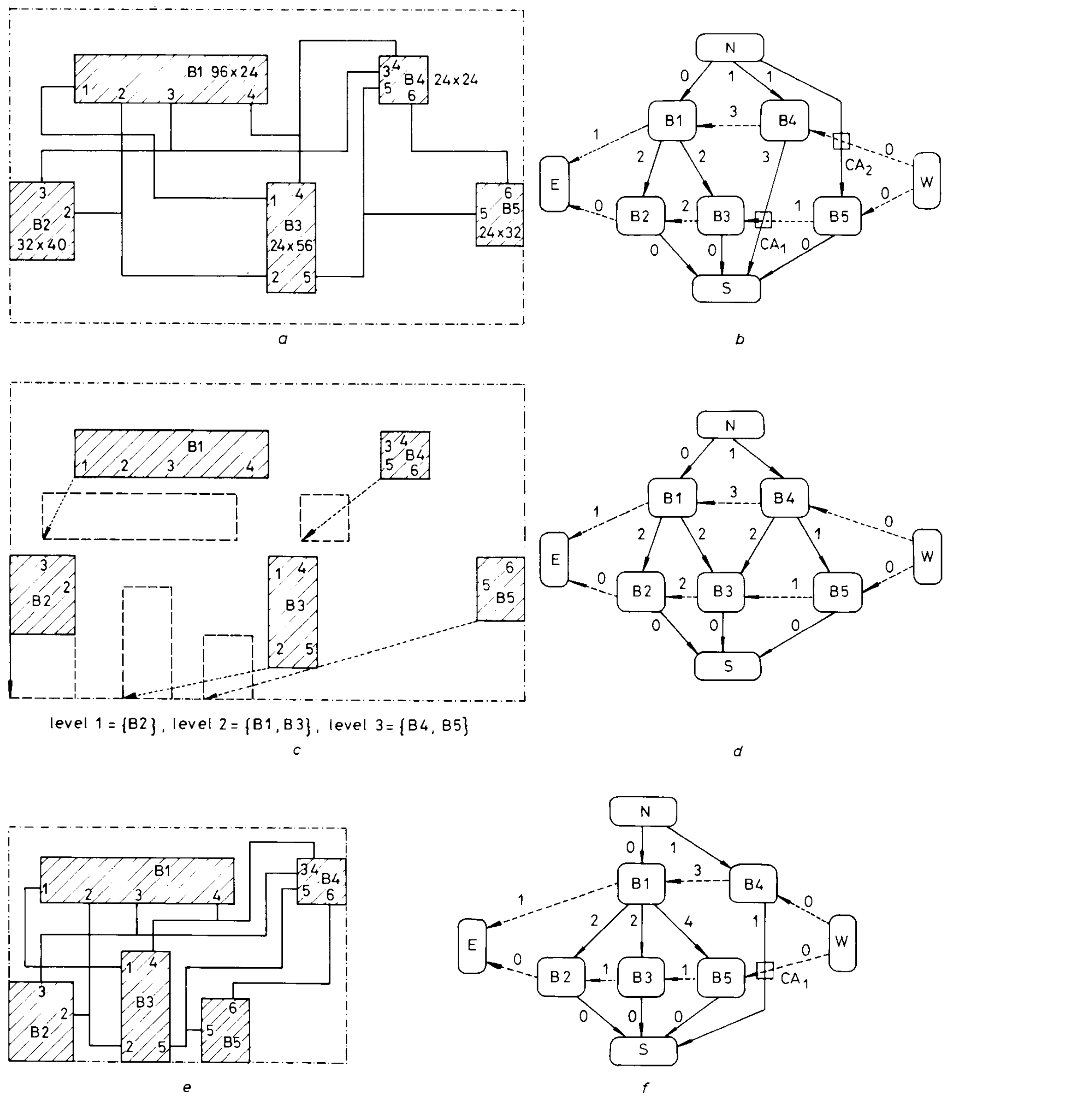
Fig. 13 *Horizontal and vertical directed graphs and the DRDG*
a Horizontal directed graph
b Vertical directed graph
c Diagonal relaxation digraph of Fig. 11*b*

at most, one of these modules should win the competitive area and change the topological relations of original module placement. For example, Fig. 14*b* is the DRDG of Fig. 14*a*, which has two competitive areas (marked by squares, *CA1* and *CA2*). Clearly, the digraph contains no channel in which capacity constraint is violated. Hence, the competitive area *CA2* can be eliminated immediately, but *CA1* cannot be, because two modules *B4* and *B5* are striving for it at the same time. This competitive area *CA1* can also be resolved by estimating the (increasing or decreasing) effect of path length caused by the moving modules on the digraph. For instance, the module *B5* may be assigned to *CA1* rather than the module *B4* because moving the module *B4* to the bottom would increase the horizontal longest path of the digraph.

Some implicit overlappings of moving modules still occur in a digraph, which would cause the topological relations of original module placement to be modified. For example, the moving modules *B4* and *B3* of Fig. 14*b* may overlap each other. These problems of implicit overlapping can also be solved by estimating the effect of path length of the moving modules on the digraph.

If a DRDG has no more competitive areas and no implicit overlapping of moving modules, the topological relation of the original module placement can be kept still, i.e. its DRDG does not change except for the number of resident tracks in each edge after every iteration.

From the above discussion, the module location refinement on the DRDG can be used to solve the explicit or implicit overlapping of moving modules, to determine the new location of each module, and to modify the topological digraph for compacting the routing areas and correcting the layout result. For instance, the modified digraph is obtained as shown in Fig. 14*c* from the DRDG of Fig. 14*b*. A heuristic method,



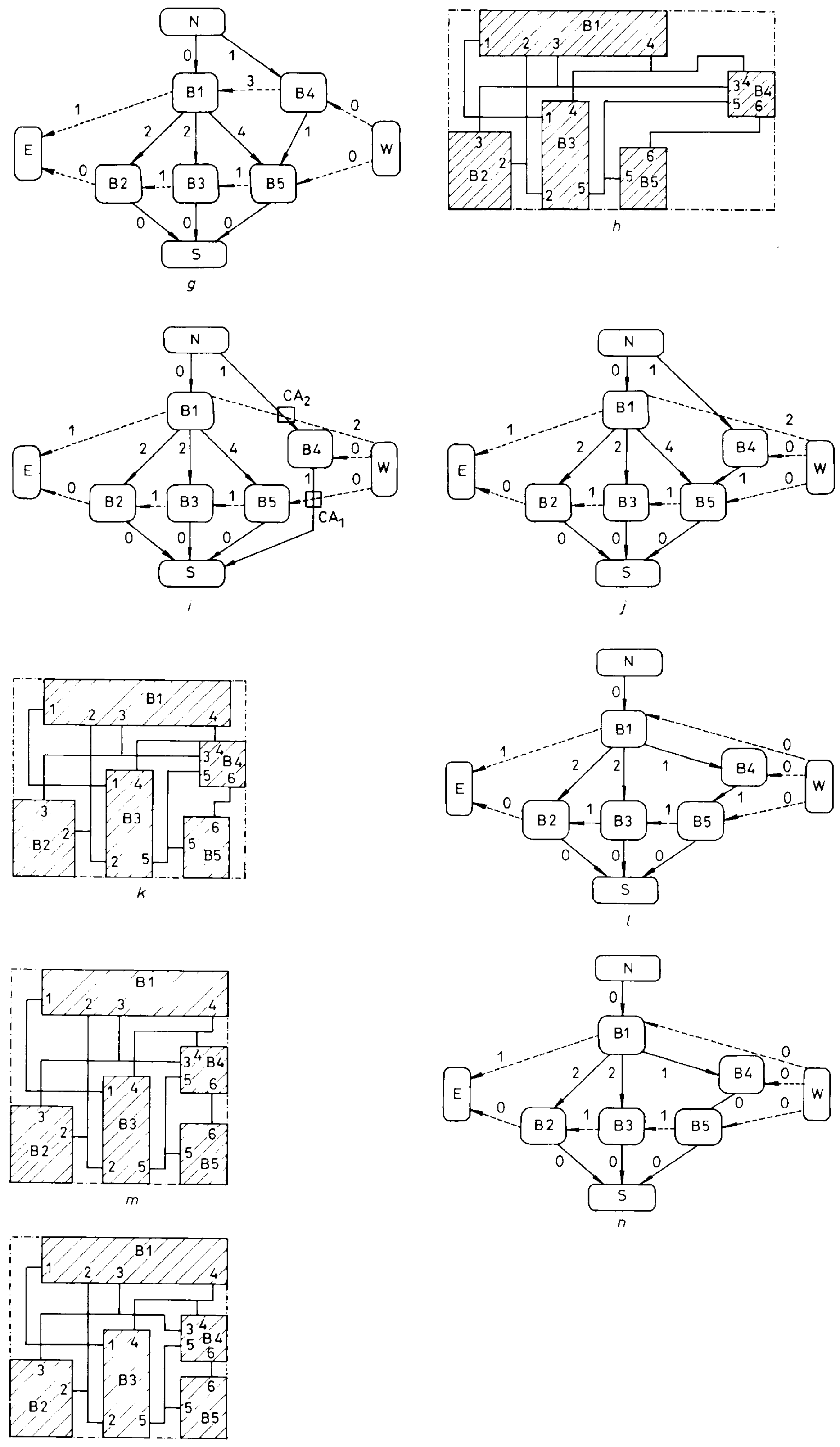


Fig. 14 *Relaxation process of an example is depicted step by step*
level 1 = {B2}, level 2 = {B1, B3}, level 3 = {B4, B5}

called diagonal wavefront movement (DWM), is investigated for these purposes. Two major rules must be held in this DWM method, as follows:

(i) The new location of a module cannot be calculated until the new location of other modules close to the left and the bottom side of this module have been determined.

(ii) Some of the modules for which new locations can be determined from eqn. 10 at the same time, are grouped in the same wavefront level, such as shown in Fig. 14*d*. The implicit or the explicit overlapping of moving modules in the same wavefront level should be considered at the same time.

With the above discussion, the module location refinement (MLR) algorithm can be described as follows:

```
MLR_Algorithm ( )
{
    Construct the DRDG from the H–V planes by the
    horizontal and vertical visibility searchings;
    DWM_procedure ( );
}

DWM_procedure ( )
{   Let $H_{min}$ and $V_{min}$ stand for the minimal current width
    and height of the main-module, respectively;
    Compute the horizontal and vertical longest path,
    called $H_{ref}$ and $V_{ref}$, respectively, from the DRDG;
    If any one of horizontal (vertical) edges violates the
    constraint of channel capacity, i.e. $H_{min} < H_{ref}$
    ($V_{min} < V_{ref}$), update $H_{min}$ ($V_{min}$);
    Find all the explicit competitive areas and all the
    moving modules sharing the competitive areas in the
    H–V planes by the horizontal and vertical visibility
    searchings;
    Then estimate whether each moving module has to be
    moved into the competitive area according to the
    values of $H_{ref}$ and $V_{ref}$ and determine the new
    topological relations of these moved modules;
    Compute the new location of the modules which
    belong to the same wavefront level in the modified
    topological digraph;
    If any implicit overlapping of a moving module
    occurs, estimate the effect of each moving module by
    calculating the values of $H_{ref}$ and $V_{ref}$, protect them
    from module overlapping, and determine the new
    topological relations of these moved modules;
}
```

Since the construction of the DRDG and the refinement of module locations are based on the tiles in the H–V planes, the time complexity of the DWM procedure is $O(cN)$, where c is a positive number and N is the number of solid tiles in the H–V plane. Therefore, the time complexity of the MLR algorithm is $O(cN)$.

Once the new locations of all the modules in the modified topological digraph have been determined, the original routing environment is also collapsed. Therefore, the channel expansion routing would be needed again, and the current result is referred by this next iteration. The whole iteration procedure will not terminate until no renewal of any module location can reduce the routing areas. By then, the global routing work is completed without violating the capacity constraint of horizontal and vertical channels. Thus, we may have a final layout with compacted routing area without performing any extra compaction work.

Fig. 14 shows the relaxing process of an example which consists of 5 modules, 6 nets, and 16 terminals. The DRDG shown in Fig. 14*b* is constructed from Fig. 14*a*. In Fig. 14*b*, there are two competitive areas (*CA1* and *CA2*) denoted by squares. Every competitive area is a spacing area shared by at least two modules. Of course, only one of the modules can win the spacing area, otherwise module overlapping will occur. Then, we compute each horizontal path from the rightmost vertex to the leftmost vertex and each vertical path from the topmost vertex to the bottommost vertex, and get the horizontal and vertical longest path length H_{ref} and V_{ref}, respectively (assume that the grid line width be 8):

$$\begin{aligned} H_{ref} &= W_g(1+1) + v_{B1}[dx] + W_g(3+1) + v_{B4}[dx] \\ &= 8(1+1) + 96 + 8(3+1) + 24 = 168 \\ V_{ref} &= v_{B1}[dy] + W_g(2+1) + v_{B3}[dy] \\ &= 24 + 8(2+1) + 56 = 104 \end{aligned} \tag{11}$$

These two competitive areas can be resolved separately; the processing order is determined according to the order of their wavefront level. Each competitive area is occupied by the module which is the most suited for it without causing any increase of the longest path length. For example, in Fig. 14*b*, the module *B5* rather than *B4* is assigned to the competitive area *CA1* by the following comparisons:

If *B4* is moved to the bottom,

$$\begin{aligned} H_x &= v_{B2}[dx] + W_g(2+1) + v_{B3}[dx] + \mathbf{W_g(2+1)} \\ &\quad + \mathbf{v_{B4}[dx]} + \mathbf{W_g(1+1)} + v_{B5}[dx] \\ &= 32 + 24 + 24 + \mathbf{24} + \mathbf{24} + \mathbf{16} + 24 \\ &= 168 \geqslant H_{ref} \quad \text{(reject)} \end{aligned} \tag{12}$$

and

If *B5* is moved to the left,

$$\begin{aligned} V_y &= W_g(1+1) + v_{B4}[dy] + \mathbf{W_g(1+1)} + \mathbf{v_{B5}[dy]} \\ &= 16 + 24 + \mathbf{16} + \mathbf{32} = 88 < V_{ref} \quad \text{(accept)} \end{aligned} \tag{13}$$

where the bold part of eqns. 12 and 13 above stand for the increasing length which is caused by moving the module *B4* or *B5*.

Consequently, a new topological digraph can be determined, as shown in Fig. 14*c*, and the new locations of all the modules are computed. Also, the new routing environment must be reconstructed to support the channel expansion routing. Fig. 14*e* shows the layout result from Fig. 14*c*. Similarly, the relaxation method above is always applied to the next iteration and attempts to compact the routing space again. Sometimes, some of the channels on which the channel capacity constraint cannot be satisfied and are required to expand for more routing space. For instance, in Fig. 14*k*, there is a channel that cannot satisfy the constraint of channel capacity in the top-right; it corresponds to the edge number (vertical edge, *B1* to *B4*) marked with the asterisk in the DRDG of Fig. 14*l*. Finally, Fig. 14*n* and *o* show the final DRDG and the compacted layout result, respectively.

6 Post refinement

From the above discussion, the layout result obtained by combining the three algorithms is likely to become the final layout result. Hence, we want to know whether the tasks of both channel expansion routing and track assignment are correct and safe, i.e. satisfy the capacity constraint of each channel and no segments overlap or touch each other.

As shown, the track assignment follows the CER algorithm; if any one of the nets is rerouted without the consideration of channel capacity constraint at the CER algorithm stage, the track assignment of some channels would not be safe (i.e. the used tracks are larger than the track density of some channels, or some modules are too crowded). This problem can be solved at the stage of refining the module locations discussed in Section 5. If all net routings satisfy the constraint of channel capacity at the CER algorithm stage, it still cannot be guaranteed that the track assignment of channels would be correct and safe without the consideration of some special cases, as discussed in Section 4.2. Fortunately, our track assignment algorithm has considered these special cases with the methods of moving modules and of inserting jog or dog-leg segments.

Also, to avoid some constraint-rule violated routing segments which exist implicitly owing to the ordering of horizontal and vertical track assignment, it needs checking to find them in the H–V planes. These constraint violated segments can be solved by the techniques of rip-up and rerouting by using jog or dog-leg segments inserting, and wire segment moving. Thus, the final layout result obtained in the H–V plane is guaranteed correct and safe.

7 Experimental results

The iterative construction approach has been implemented using standard C language and run on a SUN III/160 workstation under Berkeley 4.2 UNIX operating system. Some published examples, collected from the literature [12, 15–16], are used to measure our algorithm. Table 1 depicts these experimental results by using our approach, where only significant data are shown: the number of modules (blk), the number of nets (net), the number of terminals (term), the area of initial placement (init_area), the area of our result (area), the saving percentage of area against init_area (saving), the running time (cpu_time), and the number of iterations (iteration). Note that the init_area or area is represented by the multiplication of the width and height of the main-module. The width or height of the main-module is measured by the grid model where each grid takes eight units. From Table 1, it shows that our router can reduce routing space within a reasonable number of iterations and time.

Table 2 illustrates the comparison of the results of an X and Y compactor [1] (one-dimensional compaction beginning from the layout result of an initial placement) against our results, where the save_1 represents the saving percentage of the area of X and Y compact_result against our_result. Generally speaking, our results are always better than the results of the one-dimensional compactor. Table 3 shows the comparison of the original results against our results, where the save_2 represents the saving percentage of the area of the origin_result against our_result. The original results of the examples in Fig. 4 of Reference 16 were obtained by the method of linear programming, the example in Fig. 20 of Reference 15 was generated by the technique of digraph relaxation, and examples (Figs. 11 and 12 of Reference 12) were generated by the approach of the one- or two-dimensional compactor. From Table 3, our approach is still encouraging compared to other methods even to the two-dimensional compactor.

More example results show the effectiveness of our approach. The example expl in Table 1 is shown in Fig. 14. Another example (reconstructed by hand from the example in the literature (Fig. 20 [15], which consists of 10 modules, 33 nets, 80 terminals) in Fig. 15 shows the layout result of the initial placement, results using X and Y compaction [1], and using our method, respectively.

Table 1: Our results using the iterative construction approach

Example	Blk	Net	Term	Init_area	Our_result			
					area	saving	cpu_time	iteration
blk-1	4	5	12	304 × 272	216 × 224	41.48%	1.43 s	3
exp1	5	6	16	256 × 160	112 × 104	71.56%	3.74 s	5
[16] Fig. 4	5	12	25	432 × 336	360 × 288	28.57%	7.69 s	4
[12] Fig. 11	6	5	9	316 × 296	216 × 216	50.12%	1.04 s	3
[12] Fig. 12	11	7	18	544 × 480	368 × 336	52.64%	4.34 s	3
[15] Fig. 20	10	33	80	600 × 472	432 × 384	41.42%	56.30 s	5

Table 2: Comparison between the results of X & Y compaction and our results

Example	Blk	Net	Term	X & Y compact_result		Our_result		
				area	cpu_time	area	save_1	cpu_time
[16] Fig. 4	5	12	25	400 × 296	5.93 s	360 × 288	12.43%	7.69 s
[12] Fig. 11	6	5	9	280 × 224	2.07 s	216 × 216	25.61%	1.04 s
[12] Fig. 12	11	7	18	384 × 392	4.82 s	368 × 336	17.85%	4.34 s
[15] Fig. 20	10	33	80	448 × 416	47.2 s	432 × 384	10.98%	56.30 s

Table 3: Comparison between the original results and our results

Example	Blk	Net	Term	Origin_result		Our_result		
				area	cpu_time	area	save_2	cpu_time
[16] Fig. 4	5	12	25	360 × 296	—	360 × 288	2.70%	7.69 s
[12] Fig. 11	6	5	9	286 × 216	0.12 s (1-D)	216 × 216	25.0%	1.04 s
[12] Fig. 11	6	5	9	216 × 216	2 s (2-D)	216 × 216	0.00%	1.04 s
[12] Fig. 12	11	7	18	472 × 328	0.2 s (1-D)	368 × 336	20.1%	4.34 s
[12] Fig. 12	11	7	18	344 × 368	98 s (2-D)	368 × 336	2.32%	4.34 s
[15] Fig. 20	10	33	80	440 × 400	60 s	432 × 384	5.74%	56.30 s

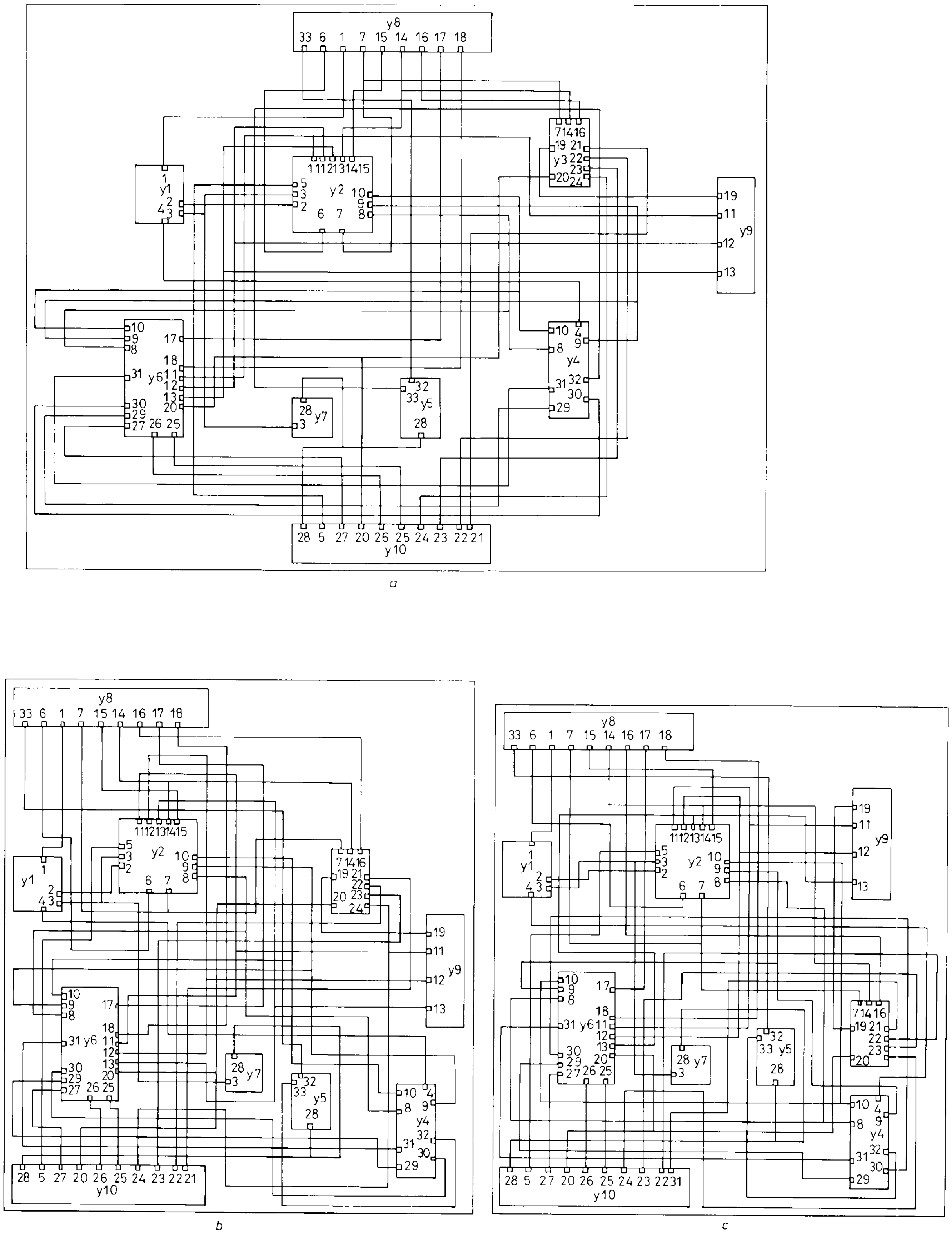


Fig. 15 *Layout example of Fig. 20 in Reference 15*

a Layout result of the initial placement
b Layout result using the method of the *X* and *Y* compaction [1]
c Our result using the iterative construction approach

8 Conclusion

A new iterative construction approach has been presented and implemented, consisting of three major algorithms: channel expansion routing, track assignment, and module location refinement. These algorithms are linked with an improved H–V model of corner-stitching, thus allowing a maximal compatitibility. This approach can guarantee a final layout with more compacted area by implicitly relaxing the module locations. Some tested results showed that our approach is better than the one-dimensional compactor and some of the two-dimensional compactors.

In future work, the module orientation and rotation [27] will be considered at the stage of module location refinement such that the final layout competes to the optimal two-dimensional compactor.

9 Acknowledgment

This work was supported by the National Science Council, Taipei, Taiwan, Republic of China under Grant NSC 78-0404-E002-46 and Grant NSC 78-0404-E002-33.

10 References

1 TSAI, C.C., and FENG, W.S.: 'HILAS — hierarchical and interactive layout system'. Proc. of Electron Devices and Material Symposium, 1987, pp. 303–308

2 BROWN, A.D.: 'Automated placement and routing', *Computer-Aided Design*, 1988, **20**, (2), pp. 39–44

3 LEE, C.Y.: 'An algorithm for path connection and its applications', *IRE Trans. Electron. Comput.*, 1961, pp. 346–365

4 HSU, C.P.: 'A new two-dimensional routing algorithm'. Proc. of 19th Design Automation Conference, 1982, pp. 46–50

5 MARGARION, A.: 'A tile-expansion router', *IEEE Trans.*, 1987, **CAD-6**, (4), pp. 507–517

6 CLOW, G.W.: 'A global routing algorithm for general cells'. Proc. of 21st Design Automation Conference, 1984, pp. 45–51

7 TSAI, C.C., CHEN, S.J., and FENG, W.S.: 'An H–V tile-expansion router'. Proc. of National Computer Conference, 1989, pp. 106–115

8 SCHIELE, W.L.: 'Improved compaction by minimized length of wires'. Proc. of 20th Design Automation Conference, 1982, pp. 121–127

9 CHO, Y.E.: 'A subjective review of compaction'. Proc. of 22nd Design Automation Conference, 1985, pp. 396–404

10 SHIN, H., SANGIOVANNI-VINCENTELLI, A.L., and SEQUIN, C.: 'Two-dimensional compaction by zone-refining'. Proc. of 23rd Design Automation Conference, 1986, pp. 115–122

11 MLYNSKI, D.D., and SUNG, C.H.: 'Layout compaction', Advances in CAD for VLSI, 4, Ohtsuki, T. editor, North Holland, 1986, pp. 199–235

12 WONG, C.K.: 'An optimal two-dimensional compaction scheme', *in* BERTOLAZZI, P., and LUCCIO, F. (Eds.) 'VLSI Algorithms and Architectures' (Elseviet Science Publishers B.V., North-Holland, 1985) pp. 205–220

13 DAI, W.M., and KUH, E.S.: 'Global spacing of building-block layout'. Proc. VLSI'87, 1987, pp. 161–173

14 FUKUI, M., YAMAMOTO, A., TAMAGUCHI, R., HAYAMA, S., and MANO, Y.: 'A block interconnection algorithm for hierarchical layout system., *IEEE Trans.*, 1987, **CAD-6**, (3), pp. 383–391

15 CIESIELSKI, M.J., and KINNEN, E.: 'Digraph relaxation for 2-dimensional placement of IC block', *IEEE Trans.*, 1987, **CAD-6**, (1), pp. 55–66

16 CIESIELSKI, M.J.: 'Two-dimensional routing for the SILC silicon compiler', *IEEE Trans.*, 1985, **CAD-4**, (3), pp. 198–203

17 DAI, W.M., and KUH, E.S.: 'A dynamic and efficient representation of building-block layout'. Proc. 24th Design Automation Conference, 1987, pp. 376–384

18 DAI, W.M., ESCHERMANN, B., KUH, E.S., and PEDRAM, M.: 'Hierarchical placement and floorplanning in BEAR', *IEEE Trans.*, 1989, **CAD-8**, (12), pp. 1335–1349

19 LI, Y.M., and TANG, P.S.: 'Global refinement for building block layout', Proc. of ICCAD, 1989, pp. 90–93

20 XIONG, X.M.: 'Two-dimensional compaction for placement refinement', Proc. of ICCAD, 1989, pp. 136–139

21 DEUTSCH, D.N.: 'A dogleg channel router'. Proc. of 13th Design Automation Conference, 1976, pp. 425–433

22 BRAUN, D., BURN, J.L., ROMEO, F., SANGIOVANNI-VINCENTELLI, A.L., MAYARAM, K., DEVADAS, S., and MA, H.K.: 'Techniques for multilayer channel routing', *IEEE Trans.*, 1988, **CAD-7**, (6), pp. 698–712

23 HAMACHI, G.T., and OUSTERHOUT, J.K.: 'A switch box router with obstacle avoidance'. Proc. of 21st Design Automation Conference, 1984, pp. 173–179

24 OUSTERHOUT, J.K.: 'Corner stitching: a data-structuring technique for VLSI layout tools', *IEEE Trans.*, 1984, **CAD-3**, (1), pp. 87–100

25 NILSSON, N.J.: Principles of Artificial Intelligence, 1980, pp. 53–94

26 GUPTA, U., LEE, D.T., and LEUNG, Y.T.: 'An optimal solution for the channel-assignment problem', *IEEE Trans.*, 1979, **CAD-28**, (11), pp. 807–810

27 RAN, L.H., and LIU, C.L.: 'Solutions to the module orientation and rotation problem by neural computation networks'. Proc. of 26th Design Automation Conf., 1989, pp. 400–405