Compaction of hierarchical cells with minimum and maximum constraints

Computing Centre Note 16

COMPACTION OF HIERARCHICAL CELLS WITH MINIMUM AND MAXIMUM CONSTRAINTS

ir. M. van der Woude and X. Timmermans

This note is an internal publication of a paper to be presented at IEEE SYMPOSIUM on Circuits and Systems, May 2-4, 1983.

April 1983

We will discuss in this paper the compaction program COMPAC which is being implemented as part of the hierarchical symbolic layout design system IDS (4).

In most cases the layout design can be done at a symbolic level. The building bricks in the symbolic design environment are:

- compound cells
  built up of elements that are instances of compound cells and/or of leave cells
- leave cells
  leave cells or sticks may be line pieces, vias, pins, transistors, etc. They are at a graphics terminal represented by an appropriate symbol.

Besides simplicity of drawing the advantages of a symbolic design are:

- easy use of CAD tools such as compactor and router
- increase in design speed, in (5) Black and Hardage report a reduction in design time of about a factor 10.

It is important that symbolic pictures look like the geometric ones. The designer must be able to predict from the symbolic layout how the ultimate mask layout will look like. Various ways of symbolic drawing with topological or virtual grids have been proposed (6), (7), (8).

However as in (3) and (9) we prefer a geometric grid for representing layouts, to meet the requirement of predictability of the geometric drawing and to make possible that compaction results can be re-edited. One grid unit (lambda) will be the characteristic minimal distance, e.g. a half minimal line spacing.

To ensure flexibility and to overcome some of the disadvantages of symbolic layout design, we will allow that both sticks and compound cells are defined geometrically and that they are in the symbolic environment represented by their domain and pins (2).

2. COMPACTION.

Compaction is expected to be very useful in combination with a symbolic layout editor, though there seem to be few applications of compaction in industry. Compaction enables the designer to input a layout with approximate placement and afterwards shift building bricks together in x and or y direction as far as design rules allow. We will restrict ourselves to discuss compaction in x-direction, since compaction in y-direction is completely similar. Various methods for compaction exist, the methods depend heavily on the symbolic grid-representation. E.g. in the virtual grid compaction method, which is used in the MULGA system (8), the minimum distance between two adjacent virtual grid lines is converted to a geometric layout. A similar method has been used by Williams (6). Other methods are shear line compaction (7) and the constraint graph method (9), (10).

The constraint graph method is very promising but seems to have been applied only to layout modules consisting of sticks, without hierarchical structure.

In the case of hierarchically structured compound cells the constraints are often two-sided or of min-max type. For example if a vertical linepiece contacts a pin area (also called terminal) of an instance of a compound cell, that line piece is allowed to shift left or right, provided the terminal area is big enough so that the contact area does not decrease. So we have two types of constraints:

- min-constraints
  stating that the distance between two elements may not decrease beyond some minimum
- min-max constraints
  stating that the distance between two elements has to be between some minimum and some maximum.

The compaction algorithm of COMPAC can handle both types of constraints.

Definitions

Before discussing the compaction algorithm we introduce some definitions.

The input of the algorithm is a compound cell from the hierarchical design database of IDS. Such a cell is built up of instances of other compound cells and of leave cells. For the compaction we will consider as elements:

- the domain of each cell instance
- the terminals at which contacts are made, either by abutment or by overlap.

A domain is a set of rectangles (at least one) which may occupy more then one layer (mask) and which covers the region occupied by a cell instance.

A terminal is the only region where a cell may have contact with the outside world it consists of at least one rectangle in at least one layer. The position of a terminal is fixed with respect to the domain of its cell.

Cells may overlap or abut only at terminals of the same layer otherwise domains of cells are separated by some minimal distance that follows from the design rules. It is a assumed that line pieces always end at terminals, in order to obtain as many degrees of freedom as possible.

An element is characterized by a local origin, the global position of which indicates the position of the element.

A feature is a subset of elements which have a fixed position with respect to each other. The elements of a compound to be compacted can be partitioned into a set of features and a set of horizontal lines. By combining the constraints of the elements of two features one can get the constraint of two features.

The min-constraint graph is an a-cyclic graph, with a source and a sink, the nodes of which correspond with the features and the edges of which are formed by the min-constraints. Min-max constraints may be indicated by bidirectional edges in the min-constraint graph thus forming the min-max constraint graph which may have cycles. A cycle of constraints is called a cyclic constraint if there is no placement of the corresponding features that satisfies all constraints.

3. PROGRAMMING STEPS FOR COMPACTION

The programming steps in COMPAC are:

3.1. Input of N layout elements.

3.2. Search for overlap.
For this search we use the algorithm for finding rectangular intersections as described in (12), which is linear "in the average" with N.

3.3. Partition elements into horizontal lines and features.
The partitioning can be performed linear in time with N by applying a breadth first search.
All features left from or on a line X = LEFTB are combined to one feature and all features right from or on a line X = RIGHTB are combined to one feature.

3.4. Determine the constraint graph.
During this step in principle each feature has to be compared with each other feature. By applying windowing techniques in combination with bin search techniques (11) the number of comparisons may be reduced dramatically (9), so that it may be performed in the average almost linear in time with the number of elements N, depending on the number of constraints.

3.5. Determine an ST ordering of the nodes of the constraint graph (12). In an ST ordening the nodes are numbered so that the edges point from lower to higher numbers.
The min max constraint may be considered bi-directional so consider first only min-constraints.
Various linear methods are known to find an ST ordering linear in time with N, e.g. the one based on depth first search. Afterwards nodes with only min-max constraints may be inserted more or less arbitrarily.

3.6. Apply the compaction algorithm, as described in the following section.

3.7. Construct from the results of the compaction a compacted compound cell.

These steps may be repeated several times for different directions (x or y) of compaction. The user can influence the results by changing the boundaries LEFTB, RIGHTB. More influence of the user may be added by allowing the user to insert user constraints, this could be programmed between step 3.2 and 3.3.

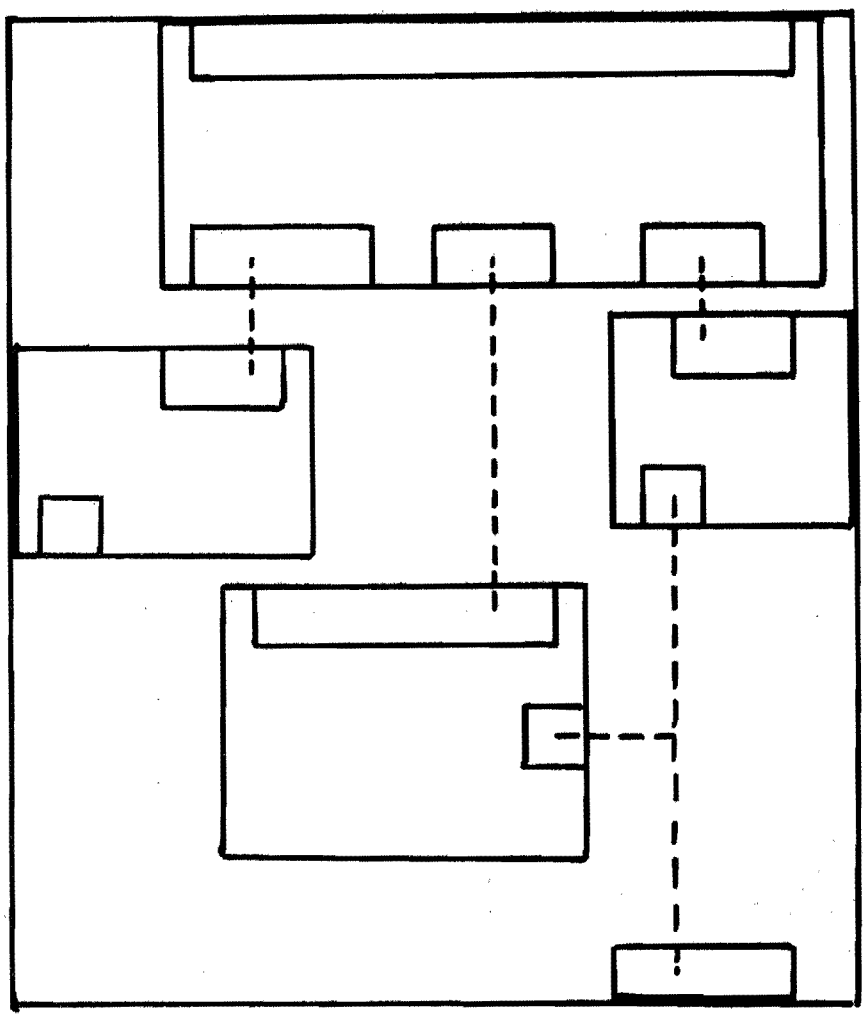

Figure 1: Original Compound Cell.

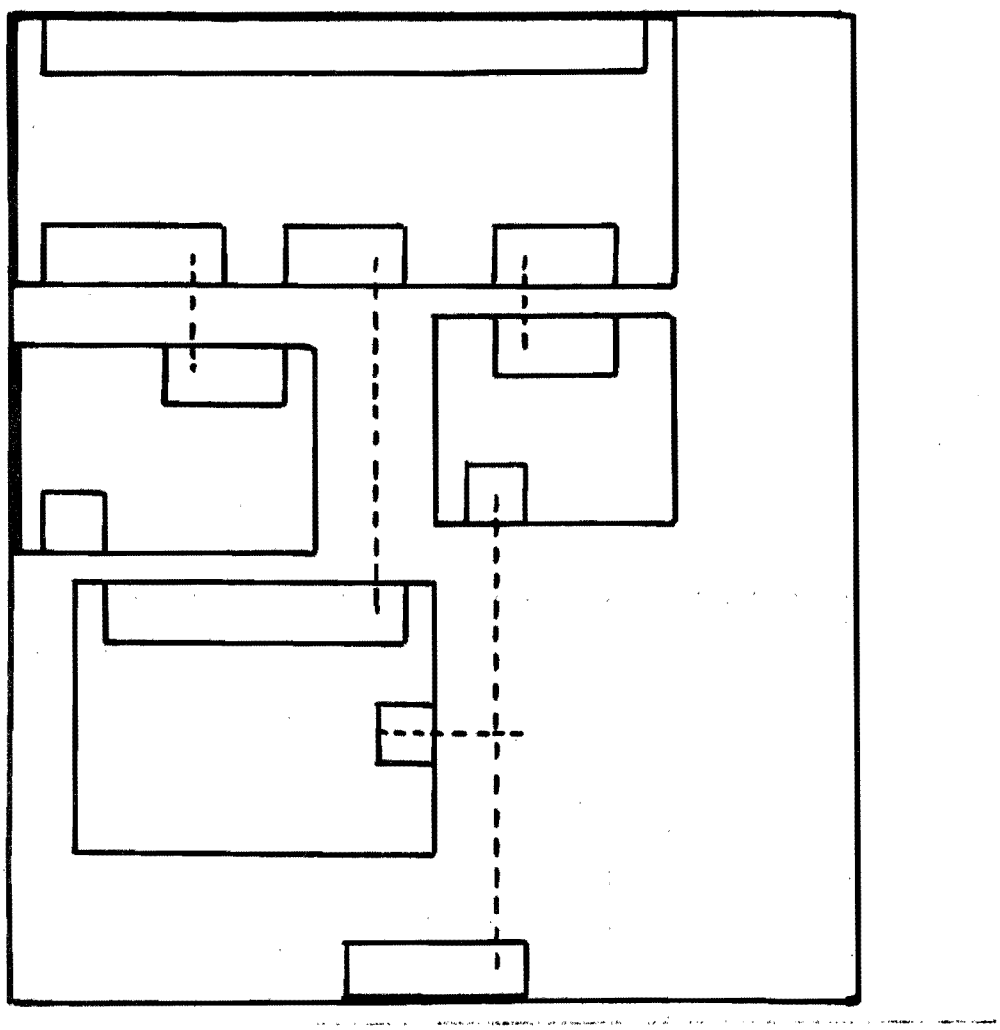

Figure 2: After X-Compaction.

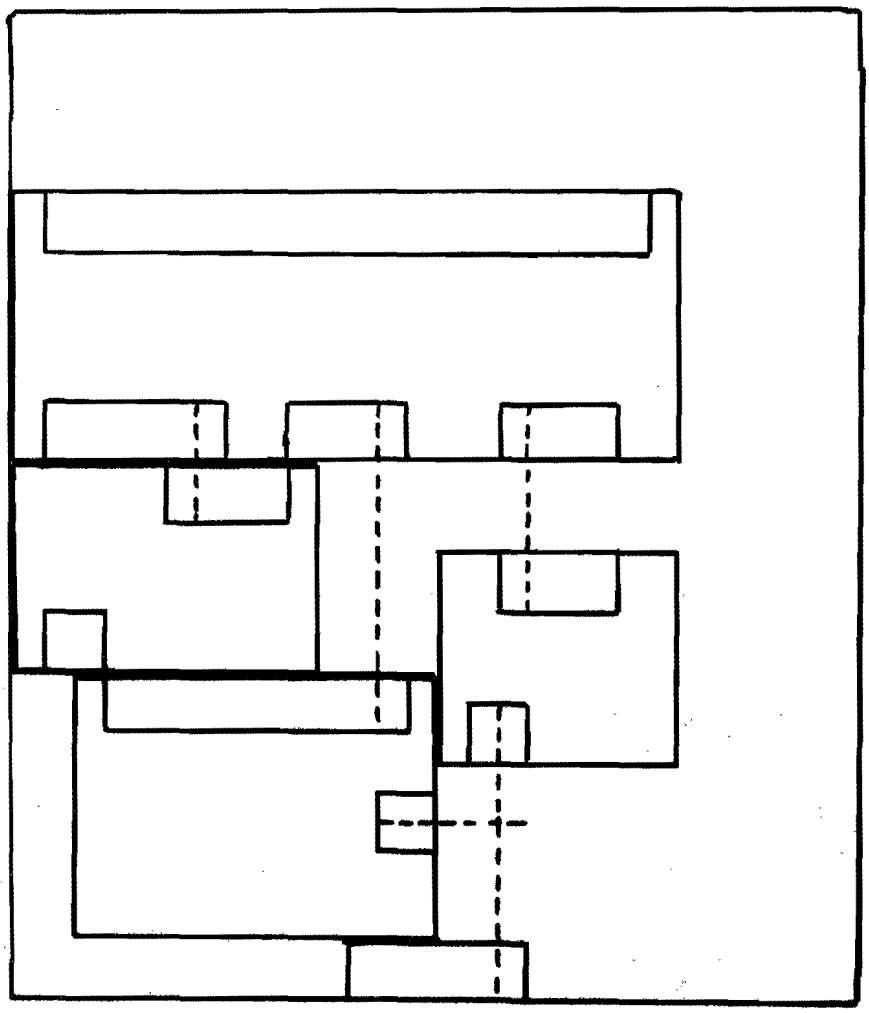

Figure 3: After X-Y Compaction.

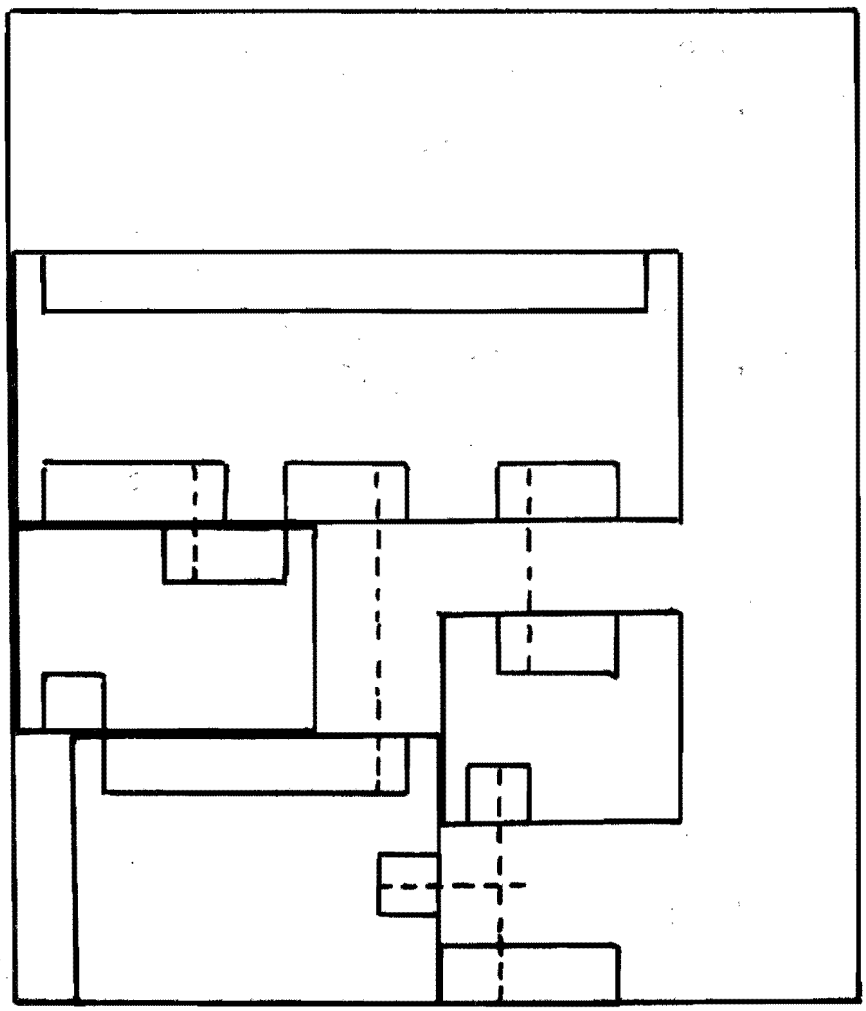

Figure 4: After Y-X Compaction.

```
COMPACTION ALGORITHM
The features are ST-numbered IF = 1,2, ..., NF
% Initialize
  for IF := 1(1) NF do FIXED[IF] := READY[IF] :=
  false; X[IF] := 0 od;
  IF := 1;
  while IF < NF
  do
    % skip IF if it is ready
    if READY[IF] then IF := IF + 1 else
    begin
        CF := IF; % CF is current feature
        if FIXED[IF] then % max constraints are satisfied
        else % check max constraints
        for each max constrained predecessor PF of CF
        do if X[CF] - X[PF] > MAXCON(CF, PF) then
           begin % pull PF
                X[PF] := X[CF] - MAXCON(CF, PF);
                READY[PF] := false; % max cons of % PF have to be
                                    % checked
                IF := min(IF, PF);
           end
        od
        % if IF < CF start back tracking
        if IF < CF then FIXED[CF] := true
        else % push successors of CF if necessary
        begin for each min constrained successor SF of CF
              do if X[SF] < X[CF] + MINCON(CF, SF)
                 then
                 begin if FIXED[SF] then exit;
                 % cyclic constraint found, stop
                 X[SF] := X[CF] + MINCON(CF, SF); % push
                 READY[SF] := false;
```

```
                end
                od;
            READY[CF] := true; FIXED[CF] := false;
            IF := IF + 1
        end
    end
od;
% end of compaction algorithm
```

Explanation of the compaction algorithm.

Each min-max constraint (IF,KF), IF < KF is considered as split into a min constraint characterized by the minimum X-coordinate MINCON(IF,KF) of the origin of KF relative to that of IF and a max constraint characterized by the maximum x-coordinate MAXCON(KF,IF) of the origin of KF relative to that of IF.

When min constraints and max constraints are considered as directed edges, then a min constraint MINCON(IF,KF) is directed from IF to KF and a max constraint MAXCON(KF,IF) is directed from KF to IF and a max constraint MAXCON(KF,IF) is directed from KF to IF. Note that in both cases IF < KF.

So if the min-max constraints are added in this way the ST numbering remains valid for all min constraints, though it had originally been constructed for only those min constraints that were not part of a min-max constraint.

We consider first the case without max constraints. We can use in this case the well-known critical path algorithm:

```
% initialize
for IF := 1(1) NF do X[IF] := 0;
IF := 1
while IF < NF
do % push successors of IF if necessary
   CF := IF;
   for each min constrained successor SF of CF
   do
     if X[SF] < X[CF] + MINCON(CF,SF) then
     X[SF] := X[CF] + MINCON(CF,SF)
   od; IF := IF + 1;
od
```

Such in algorithm has been applied in CABBAGE (10). It is linear in the number of min constraints.
Now we return to the case with max constraints present. In this case the algorithm is extended by a back tracking step, a feature CF not only pushes its min constrained successors SF, but it also pulls its max constrained predecessors PF to the right. As soon as this happens the feature CF is fixed until the back tracking has been completed. Further each pulled predecessor PF, of which all outgoing constraints were satisfied (indicated by READY PF set), has to be reconsidered. This is indicated by resetting READY PF . The same holds for features that are pushed or pulled by PF. If during the back tracking it would be necessary to push CF a cyclic constraint would have been detected. When the input was a design error free compound this of course cannot happen. If however user constraints are allowed it may occur. During back tracking features that have not moved by push or pull keep there READY value set, so they may be skipped.
The time complexity of the algorithm with back tracking becomes more than linear with the number of constraints, it is expexted that this time complexity will be almost linear when the number of max constraints is low.
In figure 1, 2, 3, and 4 this algorithm is demonstrated by compacting a set of rectangles, with minimum separation lambda.

5. CONCLUSION.

We have given the outlines of a hierarchical interactive design system and more specifically the algorithm which enables compaction of a hierarchical compound cell. Our first experiences with the design system show that the system is very flexible, due to the fact that it is at the same time leaf cell design tool and composition tool (9).

The compaction algorithm presented is of order (n), where n is the number of min constraints. If there are min-max constraints the order will increase due to the back tracking.

In (9) it is stated that there is a compaction algorithm allowing min-max constraints which is of order (n), with n likely to be the number of features, however since the algorithm is not presented it is not clear if the order (n) is also maintained when there are a lot of min-max constraints.
The compaction program COMPAC is presently under construction. As soon as experiences are available we will consider the possibility of extending the program with additional features such as stretching (8), automatic or manual jog insertion (10) and shrinking of diffusion, and poly wires by affinity (9). We expect that due to the possibility of re-editing after compaction not all these features are necessary.

6. REFERENCES.

(1) C. Niessen, "The Role of CAD Tools in VLSI Design Methodology", ESSCIRC 1981 Digest of Technical Papers, Freiburg, 22-24 Sept. 1981, pp. 75-86.

(2) K.H. Keller, A.R. Newton and S. Ellis, "A Symbolic Design System for Integrated Circuits", Proc. 19th Design Automation Conference, pp. 460-466, 1982.

(3) D.F. Bracken and W.J. Mc Calla, "An Interactive Graphics System for Structured Design of Integrated Circuits", Hewlett Packard Journal, June 1981, pp. 18-25.

(4) M. van der Woude, "IDS: an Interactive Design System for Integrated Circuits" THE Computing Centre Note 11, Eindhoven University of Technology, October 1982.

(5) K.M. Black and P.K. Hardage, "Advanced Symbolic Artwork Preparation (ASAP)", Hewlett Packard Journal, June 1981, pp. 8-10.

(6) J.D. Williams, "STICKS - A Graphical Compiler for High Level LSI Design", AFIPS Conf. Proc., Vol. 47, 1978, pp. 289-295.

(7) A.E. Dunlop, "SLIM - The Translation of Symbolic Layout into Mask Data", Proc. 17th Design Automation Conference, pp. 595-602, 1980.

(8) N. Weste, "Virtual Grid Symbolic Layout", Proc. 19th Design Automation Conference, pp. 225-233, 1981.

(9) R.C. Mosteller, "REST A Leaf Cell Design System", Silicon Structures Project Report 4317, California Institute of Technology, Pasadena, California.

(10) M.Y. Hsueh, "Symbolic Layout and Compaction of Integrated Circuits", Ph. D. Thesis University of Cal. Berkely, UCB/ERL M79/80 Memo 1979.

(11) J.L. Bentley, D. Haken and R.W. Hon, "Statistics on VLSI Designs", Report CMV-CS- 80-111, Department of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, April 17, 1980.