VLSI-Placement Based on Routing and Timing Information

J.Garbers  B.Korte  H.J.Prömel  E.Schwietzke  A.Steger
Forschungsinstitut für Diskrete Mathematik
Nassestr.2, D-5300 Bonn 1, West Germany

ABSTRACT

In this paper we propose a hierarchical placement procedure incorporating more and more detailed routing and timing information at increasing levels of the hierarchy. This procedure is based on the well-known min-cut method. A global routing and a timing analysis are computed after every cut and are used to guide the subsequent cell partitioning.

1. Introduction

The two main features of VLSI-layout are to place the components specified by the logical design on a given chip image and to realize their prescribed electrical connections by wires. An important objective is to place and to wire them in such a way that the timing is good, i.e. that the cycle-time is small.

The aim of this paper is to propose a hierarchical placement procedure incorporating more and more detailed routing and timing information at increasing levels of the hierarchy. Using this approach, we can predict routability, wire length and the cycle-time at an early stage of the layout process. Estimates of the total net length as well as of the timing given after the placement procedure are typically within a very small range of the final result.

The placement procedure is based on a successive bipartitioning algorithm (Kernighan, Lin [1970], Fiduccia, Mattheyses [1982]) that is performed in a hierarchical fashion. This successful technique for placement known as min-cut placement, was proposed by Breuer [1977] for standard cells, see also Lauther [1980], LaPotin, Director [1986], Dunlop, Kernighan [1985] and Suaris, Kedem [1988] for further developments. The idea of this method is to confine the cells to smaller and smaller regions of the chip image by repeated application of a bipartitioning procedure. Usually, this partitioning of the cells is done in such a way that cells assigned to the same region form a cluster, i.e., each net crossing a cut was counted and the number of such crossings was kept small.

Incorporating information on the routability using a global wiring procedure at each bipartitioning step was a first step towards approximating a realistic cost function for the placement step, see, e.g., Burstein, Hong, Pelavin [1983] or Suaris, Kedem [1989]. Recently, such an approach was also carried out for sea-of-cells design (Korte, Prömel, and Steger [1989]). In this paper we propose a method how after each partitioning step besides global routing information also timing information may be obtained and how this knowledge can be used to guide the subsequent cell partitioning to achieve an "optimal" placement. Coupling timing analysis with placement and routing has been proposed to influence the layout process of gate-arrays by Burstein and Youssef [1985].

The described placement procedure is part of an automatic layout system for CMOS-chips in sea-of-cells technology. It has been tested successfully for the layout of several large-scale chips (10000-20000 cells) designed in the master image design technology which was developed at the IBM Laboratories Böblingen, cf. Spruth [1989] and Koetze [1987].

This work was done within the framework of a research contract with IBM Germany. We would like to thank B. Döttling, S. Heinkele, K. Klein, J. Koehl, G. Koetze, K. Müller, R. Müller, H. Oldenburg, and R. Zühlke at the IBM Laboratories Böblingen for many valuable suggestions and for interesting and stimulating discussions, which taught us many facts about CMOS technology.

2. Obtaining global routing and timing information

In this section we describe two procedures to obtain global routing and timing information after every bipartitioning step.

2.1 Global routing. At each stage of the min-cut method the chip image is partitioned into areas \( A_i \) and each cell is assigned to exactly one such area.

Given a partition \( A \) of the chip image into areas \( A_i \) one can easily construct an adjoined global routing graph \( G(A) = (V(A), E(A)) \): Each area \( A_i \in A \) corresponds to a vertex \( v(A_i) \in V(A) \) and vice versa. Two vertices \( v(A_i) \)
and \( v(A_j) \) are connected by an edge \( e = \{v(A_i), v(A_j)\} \) if and only if the areas \( A_i \) and \( A_j \) are adjacent. Additionally, one assigns to each edge \( e \in E(A) \) a capacity \( cap(e) \) and a length \( len(e) \). Typically these values correspond to the number of wiring channels crossed by the common boundary of the corresponding areas and the Manhattan distance of the centers of gravity of these areas, respectively. Every net \( N_i \) of the netlist \( N \) induces a Steiner problem with terminal set \( T(N_i) \) in the global routing graph: \( v(A) \in T(N_i) \) if and only if \( N_i \) contains a cell \( C \) assigned to area \( A \).

The global routing problem now consists in finding a Steiner tree \( S_i \) connecting the vertices in \( T(N_i) \) for each net \( N_i \in N \). Thereby each Steiner tree should be as short as possible and no edge \( e \in E(A) \) may be used in more than \( cap(e) \) many Steiner trees.

The latter criterion is used (and necessary) to ensure routability throughout the placement procedure. By enforcing this criterion the present method differs from other approaches to combine global routing and placement (e.g. Dunlop, Kernighan [1986], Suaris, Kedem [1989]).

A detailed description of our global routing algorithm is given in Korte, Prömel, Steger [1989a]. See also Korte, Prömel, Steger [1989] for the use of the global routing during the placement algorithm.

A well-known disadvantage of a sequential routing method as we implemented is that the nets which are routed first tend to be short, while the nets which are routed last tend to be rather long, because they can only use edges \( e \) whose capacity \( cap(e) \) is not yet used up. In our algorithm this feature is used to obtain a global routing which tends to keep the cycle-time of the chip small.

To do this we associate to every net a weight. Thereby the weight of a net is proportional to the significance of a short realization of this net for the cycle-time. The net list is then sorted according to decreasing net weights. For the moment the reader may assume that these weights are provided by the logic designer. In the next section he will see how these values can be dynamically modified (starting from zero net weights for all nets) during the placement algorithm.

2.2 Timing Analysis and Evaluation. The performance of a chip is limited by the delay propagation of the signals from the chip inputs to chip outputs. Our model for timing analysis follows that used in Burstein, Youssef [1985], Hitchcock, Smith, Cheng [1982] and in Nair et. al. [1989].

We assume that a given circuit with feedback paths can be broken down into sets of closed latches such that each circuit between the various latches is pure combinational and does not contain any feedback path. This assumption permits to analyze each of these combinational circuits independently.

We use the following notations and definitions. A circuit consists of a set of cells \( C = \{C_1, \ldots, C_s\} \) and a set of nets \( N = \{N_1, \ldots, N_t\} \). Without loss of generality we assume that every cell contains exactly one output pin and that, vice versa, also every net contains exactly one output pin of a cell. We also assume that the cell set \( C \) contains all chip \( I/O \)'s. Define a directed graph \( D = (C, E) \) on the cell set: Two cells \( C_1 \) and \( C_2 \) are connected by an edge directed from \( C_1 \) to \( C_2 \) if and only if there exists a net \( N \in N \) such that \( N \) contains the output pin of \( C_1 \) and an input pin of \( C_2 \).

To each cell \( C \) (in fact, to each output pin of a cell) we associate a delay vector \( d(C) \) consisting of four values, namely the (departure) time of an up-going and of a down-going edge at the output pin and their corresponding slopes.

Originally, these values are only given for the cells corresponding to chip inputs. From there they can be followed up through the chip along the edges of the digraph described above. In particular, the delay values of a cell can be computed once the values for all predecessors are known. In our timing analysis we do a worst case (static) analysis as follows:

\[
d(C) = \max_{(C',C)\in E} f_{\text{IN}(C)}(d(C'), \text{len}(N(C)))
\]

where \( \text{len}(N(C)) \) is the length of the net \( N(C) \) containing the output pin of cell \( C \). In fact this model of propagating the delay through the cells is the one IBM uses in their Timing Analyser. See for example Spruth [1989] for technical details, in particular for a description of a function \( f_{\text{IN}(C)} \).

In the way outlined above first the clock signals are traced from the chip inputs to their terminals on the latches. In a second step all combinational paths are traced between chip input/latch output pins and chip output/latch input pins. (Observe that the delay values of the latch output pins are given by the values of the incident clock net.)

The arrival time of the up-going and down-going edges of the clock signal at a latch plus the cycle-time define the time when the data signal of the combinational path has to arrive at the latch. The difference between this required time of arrival \( t_u(L) \) and the actual arrival time \( t_a(L) \) at a latch \( L \) is called the slack:

\[
s(L) = t_u(L) - t_a(L).
\]

These slack values can be computed in linear time, i.e. in \( O(|C| + |N|) \), for all latches and chip output cells (for the chip output cells we assume that the required times of arrival are pregiven). To simplify notation we denote this subset of the cell set \( C \) by \( S \).

Now consider \( \min_{L\in S} s(L) \) : a positive value means that the current cycle-time is (or will probably be, if the timing is analysed during the placement phase) fulfilled by this physical layout, while a negative value indicates that the layout violates the timing conditions. In addition, a large positive value indicates that the cycle-time can be further reduced (by about this value). Hence, our goal is to
maximize \( \min_{L \in \mathcal{S}} s(L) \).

Recall that after each cut of the min-cut algorithm we compute a global routing as described above. In particular this provides us with an expected net length for every net. These net lengths are subsequently used to perform a timing analysis. The main idea of incorporating the obtained timing information in the actual cell bipartitioning step (cf. next section) is to modify the net weights \( w(N) \) associated to every net \( N \).

In particular, increasing the weight of some nets should lead to a shorter realisation of these nets and thus should increase the minimal slack. A straightforward strategy which is easy to implement is the following: Consider all latches/chip outputs which achieve the minimal slack or are close to it, i.e. set
\[
S^* = \{ L \in \mathcal{S} \mid s(L) \leq \min_{L' \in \mathcal{S}} s(L') + \delta \}
\]
for some small constant \( \delta \). For all elements \( L \in S^* \) trace the critical path from \( L \) backwards to the startpoint of the combinational path ending at \( L \). This can be done by recursively taking the one predecessor which achieved the maximum value during the delay computation. This procedure yields a set \( N^* \) of nets contained in critical paths.

We now change the net weights as follows:
\[
w(N) := \begin{cases} w(N) + 1 & N \in N^*, \\ w(N) & \text{otherwise}. \end{cases}
\]

Though this strategy is quite simple and one might easily think of more complicated ones, the results are already very promising.

3. Cell partitioning

In this section we discuss the actual partitioning of the cells. Let \( A_1 \) and \( A_2 \) be the areas originating from an area \( A_0 \) by the \( i \)-th cut. With \( s(A) \) we denote the size of an area \( A \) and with \( C(A) \) the set of cells contained in \( A \). The cell density \( \text{dena}(A) \) of an area \( A \) is given by \( \text{dena}(A) = \sum_{C \in C(A)} \frac{w(C)}{s(A)} \), where \( w(C) \) denotes the size of a cell \( C \).

A balanced cell partition (short: partition) of \( C(A_0) \) with respect to \( A_1 \) and \( A_2 \) is a pair \((C(A_1), C(A_2))\) such that \( C(A_1) \cup C(A_2) = C(A_0) \) and \( C(A_1) \cap C(A_2) = \emptyset \) and such that
\[
|\text{dena}(A_1) - \text{dena}(A_2)| \leq \frac{\max\{s(C) \mid C \in C(A_0)\}}{\min\{s(A_1), s(A_2)\}}.
\]

The "quality" of a partition \((C(A_1), C(A_2))\) is generally measured by the value of an objective function \( f(C(A_1), C(A_2)) \). Given such a function the cell partition problem is defined as follows:

Given areas \( A_0, A_1 \) and \( A_2 \) find among all partitions \((C(A_1), C(A_2))\) a partition \((C^*(A_1), C^*(A_2))\) such that \( f(C^*(A_1), C^*(A_2)) \) is minimal.

The objective function we use is
\[
f(C(A_1), C(A_2)) = \sum_{C \in C(A_0)} \nu(N),
\]
where \( \nu(N) \) denotes the contribution of net \( N \). It depends on the distribution of the cells of \( N \) (denoted by \( C(N) \)) contained in \( A_0 \) (denoted by \( l_c(N) = C(N) \cap C(A_0) \)), read: local cells of \( N \) to the areas \( A_1 \) and \( A_2 \), on the global routing information and on the timing evaluation.

The precise definition of \( \nu(N) \) is \( \nu(N) = g(N) \cdot h(N) \) with
\[
g(N) = \frac{c(w(N)) \cdot \max\{1, \frac{\text{len}(N)}{q(w(N))}\}}{}
\]
where \( c(w) \) is a monotone increasing function and \( g(w) \) is a monotone decreasing function in \( w \), and \( \text{len}(N) \) denotes the current length of net \( N \) (provided by the global routing). Finally, the function \( h \) is given by
\[
h(N) = \begin{cases} 1 & \text{if } C(N) \cap C(A_i) \neq \emptyset \text{ and } C(N) \cap C(A_j) \neq \emptyset, \\ \max_{e_{ij} \in \text{GR}(N)} \frac{c(w(e_{ij}))}{\text{len}(e_{ij})} & \text{if } C(N) \cap C(A_i) \neq \emptyset \text{ and } C(N) \cap C(A_j) = \emptyset, \\ \max_{e_{ij} \in \text{GR}(N)} \frac{c(w(e_{ij}))}{\text{len}(e_{ij})} & \text{if } C(N) \cap C(A_i) = \emptyset \text{ and } C(N) \cap C(A_j) \neq \emptyset, \\ \max_{e_{ij} \in \text{GR}(N)} \frac{c(w(e_{ij}))}{\text{len}(e_{ij})} & \text{if } C(N) \cap C(A_i) = \emptyset \text{ and } C(N) \cap C(A_j) = \emptyset, \end{cases}
\]
where \( \text{GR}(N) \) denotes the global routing of net \( N \), i.e. \( \text{GR}(N) \) is a Steiner tree in \((V(A_i), E(A_i))\) for \( T(N) \). Additionally, \( e_1 \) and \( e_2 \) denote the edges of the global routing graph resulting from an edge \( e \) in the \( i \)-th cut.

Observe that for each cut the time complexity for computing the net values \( \nu(N) \) for all nets is bounded by \( O(|A| \cdot |N|) \).

We now turn to the methods to find a solution for the partitioning problem with respect to this objective function \( f \). This is accomplished in two phases. In a first step we construct a "reasonable" initial solution, in a second step this solution is improved by an iterative improvement method. In this note we discuss only the improvement method.

Our improvement algorithm is essentially based on the well known Kernighan-Lin-exchange-heuristic. Just as the KL-algorithm we try to improve our initial solution in several iterations. But instead of exchanging pairs of cells we will shift only one cell at a time. This has been done for two reasons. Firstly, we have cells with different cell sizes. Thus it is impossible to exchange an arbitrary pair of cells in each step, because this would result in too varying cell densities of our areas. Recall, that the cell densities of our areas should be invariants. Secondly, the running time aspect: To find the best cell for exchange can be done in \( O(|C(A_0)|^2) \) instead of \( O(|C(A_0)|^3) \) for pairs of cells. Similar considerations can be found in Fiduccia, Mattheyses [1982].
Further differences to the KL-algorithm are the objective function \( f(C(A_1), C(A_2)) \) and the (local) exchange evaluation function \( f_c(C) \) which is used to identify the next cell to be shifted. (A precise definition of \( f_c(C) \) is given below.)

Now we give a short description of a single iteration of our algorithm: We start with a partition \((C(A_1), C(A_2))\), and try to improve it by subsequent single swaps of cells. But within an iteration each cell can only be shifted once.

That means a cell which is already exchanged during the current iteration will be temporarily fixed (marked) until the next iteration starts. Only unmarked cells will be exchanged. Among all these cells we search for a cell \( C \) which satisfies the density exchange condition and which has maximum value \( f_c(C) \), i.e. which is moved to an area \( A_i \) such that \( \text{dens}(A_i) \leq \text{dens}(A_{i-1}) \).

A disadvantage of that function is that only the real change of the objective function \( f(C(A_1), C(A_2)) \) will be considered. That means that only those nets are taken into account which local cells \( ic(N) \) all lie in the same area when cell \( C \) is exchanged or which local cells will be separated by shifting the cell \( C \). All other nets with intermediate cell distributions are ignored.

We now introduce an evaluation function which tries to get rid of this disadvantage by looking ahead. Denote by \( fis(N, A_i) \) the number of local cells of net \( N \) which are already exchanged and fixed to area \( A_i \), during the current iteration of the improvement algorithm. Recall, that by definition of \( \nu(N) \) the condition

\[
0 \leq \nu(N, q_i) \leq \nu(N, q_0), \quad i = 1, 2
\]

is valid. Denote by \( tfp(N, C) = \int v(N, q_i(C)) - \nu(N, q_0(C)) \mid \text{ic}(N) \mid \nu(N, q_i)^2 \) the turning point of cell \( C \) relative to net \( N \). The turning point will be used to define the tendency of cell \( C \) with regard to net \( N \). The improved evaluation function, subsequently called Cell-Distribution (CD)-function \( fn(N, C) \), is given by

\[
\begin{align*}
\nu(N, q_i(C)) &= \nu(N, q_0(C)) \quad \text{if } fis(N, A_i) = 0 \\
\nu(N, q_i(C)) &= \nu(N, q_0(C)) \quad \text{if } fis(N, A_i) > 0 \\
\nu(N, q_i(C)) &= \nu(N, q_0(C)) \quad \text{if } fis(N, A_i) \leq \text{fp}(C, G)) \\
\nu(N, q_i(C)) &= \nu(N, q_0(C)) \quad \text{if } fis(N, A_i) > 0 \\
\nu(N, q_i(C)) &= \nu(N, q_0(C)) \quad \text{if } fis(N, A_i) \leq \text{fp}(C, G)) \\
\nu(N, q_i(C)) &= \nu(N, q_0(C)) \quad \text{otherwise.}
\end{align*}
\]
This function takes into consideration the net value \( v(N) \), already performed exchanges, and the distribution of the local cells of net \( N \) with regard to areas \( A_1 \) and \( A_2 \).

The time complexity of the improvement algorithm for performing a single iteration is bounded by \( O(|N| \cdot |C(A_0)|^2) \). Applied to typical VLSI-circuits 3-6 iterations usually suffice to find a locally optimal partition.

4. Results

The described placement algorithm is implemented in PASCAL on an IBM 3081-K under VM/CMS. It contains about 50,000 lines of code.

In table 1 we present the results of the placement algorithm on the first two cuts for a chip containing about 13,000 cells and more than 16,000 nets.

<table>
<thead>
<tr>
<th>cut</th>
<th>start heuristic</th>
<th>cost of start solution</th>
<th>cost of final solution</th>
<th>number of iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>random</td>
<td>9697.00</td>
<td>1835.00</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>constructive</td>
<td>420.00</td>
<td>379.00</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>random</td>
<td>4810.00</td>
<td>681.45</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>constructive</td>
<td>566.80</td>
<td>533.00</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 2 studies the influence of the interaction of the placement with the global routing and the timing analysis. These data were calculated for a chip with roughly 8,000 cells and almost 10,000 nets.

<table>
<thead>
<tr>
<th>Global Routing</th>
<th>Timing Analysis</th>
<th>global routing</th>
<th>minimal slack</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>net length</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>17.52</td>
<td>2.7</td>
</tr>
<tr>
<td>x</td>
<td>-</td>
<td>13.09</td>
<td>4.5</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>13.10</td>
<td>5.3</td>
</tr>
<tr>
<td>net length</td>
<td>0</td>
<td>-</td>
<td>7.4</td>
</tr>
</tbody>
</table>

References