M. Jünger, A. Martin, G. Reinelt, R. Weismantel

Quadratic 0/1 Optimization and a Decomposition Approach for the Placement of Electronic Circuits

Revised Version of April 92

Preprint SC 92–10 (April 1992)

Abstract

The placement problem in the layout design of electronic circuits consists of finding a non-overlapping assignment of rectangular cells to positions on the chip so that wireability is guaranteed and certain technical constraints are met. This problem can be modelled as a quadratic 0/1-program subject to linear constraints. We will present a decomposition approach to the placement problem and give results about $\mathcal{NP}$-hardness and the existence of ε-approximative algorithms for the involved optimization problems. A graphtheoretic formulation of these problems will enable us to develop approximative algorithms. Finally we will present details of the implementation of our approach and compare it to industrial state of the art placement routines.

Keywords

Quadratic 0/1 optimization, Computational Complexity, VLSI-Design

1. Introduction

The layout problem for integrated electronic circuits is the following task: We are given a finite set B of cells in which each **cell** $b \in B$ has a finite number of rectangular realizations which are characterized by their widths and heights, the positions of their electrical terminals (pins) and their electrical properties like switching speed, etc. Furthermore we have a set of nets $N \subseteq 2^B$, and each **net** $\{b_1, b_2, \ldots, b_k\} \in N$ specifies that certain pins of the cells $b_1, b_2, \ldots, b_k$ have to be electrically connected. We have to determine a nonoverlapping assignment of the cells to the plane and a realization of the nets such that certain criteria like minimal area of the smallest rectangle containing all cells or minimal switching time of the circuit are satisfied. In practice, there are usually many partially contradicting such criteria, and their relative weights have to be controlled by parameters. Often, the rectangular area in which the circuit has to be realized is fixed, or maximal switching times are demanded, such that the problem becomes a feasibility rather than an optimization problem. The layout problem presents one of the major challenges of modern industry to mathematics and computer science.

Any reasonably precise version of the layout problem is $\mathcal{NP}$-hard, even very simple ones. Moreover, most real world problem instances are very large, so that today's algorithmic knowledge makes it very improbable that they can be solved to optimality. Therefore, usually heuristic decomposition into subproblems is applied, and the subproblems are treated with heuristics and sometimes exact algorithms. Usually, the layout problem is decomposed into placement of cells, then wiring and finally compaction. This process is iterated with different parameters if the final result is not satisfactory. An excellent treatment of this subject can be found in LENGAUER (1990).

In this paper we deal with the first phase of the process outlined above, namely the placement of cells in a common style, the sea-of-cells (or sea-of-gates) layout of VLSI-chips.

In Section 2 we give a precise formulation of the placement problem in the sea-of-cells layout style. Then we introduce a new optimization model for this problem based on a quadratic 0-1 programming problem with special linear constraints.

The complexity of various versions of this optimization problem is considered in Section 3. We show that even simple versions are not only $\mathcal{NP}$-hard, they cannot even be solved approximately in polynomial time unless $\mathcal{P}=\mathcal{NP}$.

Our complexity results indicate that we cannot expect to find satisfactory solutions to reasonably sized instances of the placement problem directly by our method. Therefore, Section 4 deals with a method which decomposes an instance of the problem recursively into several small instances in a novel way. Our decomposition scheme keeps a global view of the entire problem instance at all stages. This feature is not shared by commonly used decomposition methods like e. g. "min-cut-tree" decomposition.

In Section 5 we point out the equivalence of our optimization model to a special clique problem in a graph. This formulation gives rise to several heuristics for small sized problem instances as they have to be solved during the decomposition process.

The computation of good solutions to small instances of our optimization problem still has to deal with a considerable amount of data which defines the objective function of the quadratic 0-1 programming problem. Section 6 presents several implementation features of our experimental software which are crucial for making our program run reasonably fast on real world instances of the placement problem.

Finally, Section 7 gives computational results which show that our approach is competitive with commonly used software.

2. The quadratic 0/1 model

The layout style we consider here is usually called "channelless gate array"-, "sea of gates"- or "sea-of-cells"-layout style. An integrated circuit in this technology consists of a rectangular array of **base cells** without predefined wiring channels (the **master**) plus an outer part which consists of two rows and two columns of base cells at the borders of the master (**pad cells**), cf. Figure 1.

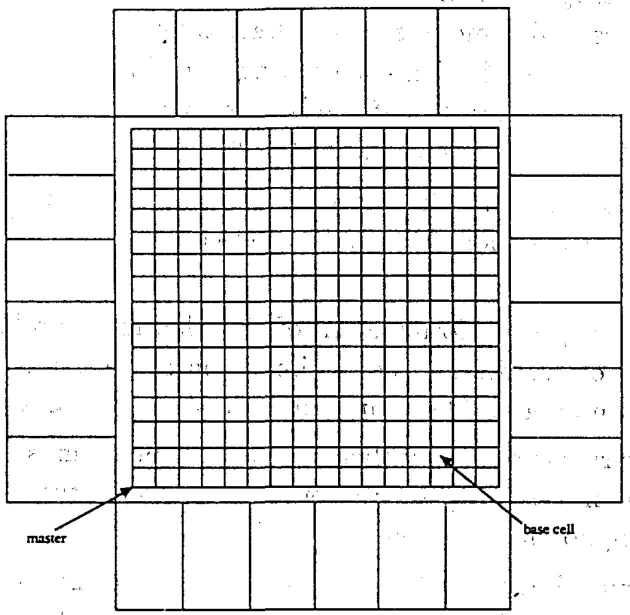

Figure 1

Any **logic cell** (or simply **cell**) to be placed on the master covers a rectangular array of base cells with uniquely defined dimensions. Pad cells are preassigned to one of the four outer parts, and cover one base cell. Inter cell wiring takes place on uncovered base cells or unused tracks of cells.

The placement problem consists of finding an assignment of the cells to disjoint areas of base cells of appropriate dimensions such that certain side constraints like wireability with minimum total wire length are met. Our approach can deal with more than one possibility of the dimensions for a logic cell, so it is also suitable as a component of a general floor planning technique, for details see WEISMANTEL (1992).

We say that cell i is **assigned** to base cell k if cell i is placed on the master so that its lower left corner coincides with base cell k. A base cell k is called **feasible** for a cell i if cell i fits on the master when assigned to base cell k.

Let n denote the number of logic cells and b the number of base cells of the master. We introduce variables x_{ik} for $i = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, b$ such that:

$$x_{ik} = \begin{cases} 1, & \text{if logic cell } i \text{ is assigned to feasible base cell } k \ , \\ 0, & \text{otherwise.} \end{cases}$$

Each logic cell must be assigned to exactly one of the b base cells, so the following set of equations has to be satisfied:

$$\sum_{k=1}^{b} x_{ik} = 1 \quad \text{for all } i = 1, 2, \ldots, n.$$

Since the wireability condition is difficult to model it is replaced in practice by an objective function estimating the total wiring length. By simply counting the number of nets connecting both cells i and j the wiring length of nets with cardinality more than two will be overestimated. To this end we calculate **affinity coefficients** $c_{ij} \geq 0$ between i and j according to the formula

$$c_{ij} = \sum_{\substack{\text{net } t \text{ connects} \\ i \text{ and } j}} \frac{1}{(\text{cardinality of } t) - 1}.$$

$d_{ik,jl}$ denotes the Manhattan distance between cells i and j when assigned to base cells k and l respectively. The Manhattan distance between cells i and j when assigned to base cells k and l, respectively, is the sum of the shortest distances in horizontal and vertical direction between any two points on the boundary of i and j when assigned to base cells k and l, respectively. For reasons that will become clear later we refrain from introducing constraints that guarantee that no two cells of a placement overlap. We rather treat overlappings by modifying the objective function appropriately. Let $o_{ik,jl} \geq 0$ be the number of overlapping units (i. e. the number of common base cells), if cells i and j are assigned to base cells k and l respectively. Finally, let R denote the set of pad cells and $p(r)$ the (predefined) location of pad cell $r \in R$. We define a cost matrix Q by setting

$$q_{ik,jl} = c_{ij} \cdot d_{ik,jl} + \lambda o_{ik,jl} \quad \text{for all } i,j = 1,2,\ldots,n, \ i \neq j \text{ and } k,l = 1,2,\ldots,b,$$
$$q_{ikil} = 0 \quad \text{for all } i = 1,2,\ldots,n, \ k,l = 1,2,\ldots,m,$$

where $\lambda \geq 0$ is a penalty parameter. With these definitions our model for the placement problem can now be stated as follows:

$$(2.1) \qquad \begin{aligned} &\min \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{l=1}^{m} q_{ik,jl}x_{ik}x_{jl} + \sum_{r\in R}\sum_{i=1}^{n}\sum_{k=1}^{m} c_{ir}d_{ik,r,p(r)}x_{ik} \\ &\sum_{k=1}^{b} x_{ik} = 1 \quad \text{for all } i = 1,2,\dots,n, \\ &x_{ik} \in \{0,1\} \quad \text{for all } i = 1,2,\dots,n, \text{ and for all } k = 1,2,\dots,b. \end{aligned}$$

Since $x_{ik}^2 = x_{ik}$ for all $i = 1,2,\dots,n$ and $k = 1,2,\dots,m$ we can define a matrix Q' via

$$\begin{aligned} q'_{ikjl} &= q_{ik,jl} \quad \text{for all } i,j = 1,2,\dots,n,\ i \neq j,\ k,l = 1,2,\dots,m, \\ q'_{ik,ik} &= \sum_{r\in R} c_{ir}d_{ik,r,p(r)} \quad \text{for all } i = 1,2,\dots,n,\ k = 1,2,\dots,m, \end{aligned}$$

such that

$$(2.2) \qquad \begin{aligned} &\min x^T Q' x \\ &\sum_{k=1}^{b} x_{ik} = 1 \quad \text{for all } i = 1,2,\dots,n, \\ &x_{ik} \in \{0,1\} \quad \text{for all } i = 1,2,\dots,n, \text{ and for all } k = 1,2,\dots,b. \end{aligned}$$

is equivalent to (2.1). In the remainder of this paper the quadratic optimization problem (2.2) will be denoted by (P).

In addition to the general problem (P) we will consider the variant (P_m) where we have for each cell a fixed number m of feasible base cells. This variant of (P) will play an important role as a subproblem coming up during the hierarchical decomposition process to be described in Section 4. For ease of notation we assume that the feasible base cells for cell i are ordered and we use the interpretation that $x_{ik} = 1$ if cell i is assigned to its k-th feasible base cell. For convenience, we will still say that cell i is assigned to base cell k. The reader should keep this simplification in mind.

$$(P_m) \qquad \begin{aligned} &\min x^T Q' x \\ &\sum_{k=1}^{m} x_{ik} = 1 \quad \text{for all } i = 1,2,\dots,n, \\ &x_{ik} \in \{0,1\} \quad \text{for all } i = 1,2,\dots,n, \text{ and for all } k = 1,2,\dots,m. \end{aligned}$$

Analogously we define problems (P^+) and (P_m^+) if we require in addition that $Q' \geq 0$.

3. Computational complexity of the placement model

In this section we deal with the complexity of the problems (P) and (P_m) and their variants for nonnegative Q. We will show that they belong to the class of $\mathcal{NP}$-hard problems. In addition, we determine the complexity of finding approximative solutions for these problems.

(3.1) Definition *For any $\varepsilon > 0$ an algorithm A is said to be an* **ε-approximative algorithm** *for a minimization problem if, for any instance I of the problem, A satisfies*

$$c_A^I \leq (1+\varepsilon)c_{\text{opt}}^I,$$

where $c_A^I \geq 0$ denotes the objective function value of a feasible solution obtained by A and $c_{\text{opt}}^I \geq 0$ denotes the objective function value of an optimal solution for instance I. The number ε is the **performance guarantee** *of A (*Garey & Johnson (1979)*).*

In the following we will show that problems (P), (P^+), (P_m) and (P_m^+), $m \geq 2$, are $\mathcal{NP}$-hard. In addition, there do not exist polynomial time ε-approximative algorithms for these problems for any $\varepsilon > 0$ unless $\mathcal{P} = \mathcal{NP}$.

(3.2) Theorem. *Let $m \geq 3$ fixed. There exists no polynomial time ε-approximative algorithm for (P_m^+) for any $\varepsilon > 0$ unless $\mathcal{P} = \mathcal{NP}$.*

Proof. Suppose that for some $\varepsilon > 0$ algorithm A is a polynomial time ε-approximative algorithm for (P_m^+).
The m-colorability problem consists of deciding for a given loopless graph $G = (V, E)$ with node set V and edge set E whether there exists a function $\mu : V \to \{1, 2, \ldots, m\}$ such that $\mu(v) \neq \mu(w)$ for each edge $e = (v, w) \in E$. This problem is $\mathcal{NP}$-complete (Karp (1972)).
Let $n = |V|$ and represent any m-coloring of G bijectively via

$$x_{ik} = \begin{cases} 1, & \text{if } \mu(i) = k, \\ 0, & \text{otherwise.} \end{cases}$$

Furthermore let

$$M = \varepsilon n^2 + 2$$

and define for $k, l \in \{1, 2, \ldots, m\}$ and $i, j \in \{1, 2, \ldots, n\}$ the matrix $Q \in \mathbf{R}^{mn \times mn}$ by

$$q_{ik,jl} = \begin{cases} M, & \text{if } (i,j) \in E, \text{ and } k = l, \\ 1, & \text{otherwise.} \end{cases}$$

Obviously, G is m-colorable if and only if (P_m^+) with parameters n and Q has a feasible solution with objective function value $c_{\text{opt}} = n^2$.

Now apply algorithm A to (P_m^+). If G has an m-coloring, then A produces a feasible solution x^A with objective function value $c_A \leq (1+\varepsilon)n^2 < (n^2-1) + M$. Thus for any matrix entry $q_{ik,jl} = M$ the values x_{ik}^A and x_{jl}^A cannot both be 1. Therefore, $\mu(i) = k$ if and only if $x_{ik}^A = 1$ represents an m-coloring of G. On the other hand, if G has no m-coloring, A produces a feasible solution x^A with objective function value $c_A \geq c_{\text{opt}} \geq (n^2-1) + M$. Therefore, we can use algorithm A to decide in polynomial time whether G is m-colorable, which is impossible unless $\mathcal{P} = \mathcal{NP}$. □

(3.3) Corollary. *Problems (P_m^+), $m \geq 3$ are $\mathcal{NP}$-hard.*

□

(3.4) Corollary. *Problems (P_m), $m \geq 3$, (P) and (P^+) are $\mathcal{NP}$-hard. Furthermore, there does not exist a polynomial time ε-approximative algorithm for (P_m), $m \geq 3$, (P) and (P^+) for any $\varepsilon > 0$ unless $\mathcal{P} = \mathcal{NP}$.*

□

(3.5) Theorem. *There does not exist a polynomial time ε-approximative algorithm for (P_2) for any $\varepsilon > 0$ unless $\mathcal{P} = \mathcal{NP}$.*

Proof. Suppose that for some $\varepsilon > 0$ algorithm A is a polynomial time ε-approximative algorithm for (P_2).

The 2-partitioning problem consists of deciding for n given integers $r_1, r_2, \dots, r_n$ whether there exist two disjoint subsets $I_1, I_2 \subset \{1,2,\dots,n\}$ such that $I_1 \cup I_2 = \{1,2,\dots,n\}$ and $\sum_{i\in I_1} r_i = \sum_{i\in I_2} r_i$. This problem is known to be $\mathcal{NP}$-complete (Karp (1972)).

Let $M = \varepsilon + 2$. We introduce variables x_{i1} and x_{i2} for all $i = 1,2,\dots,n$ with the interpretation

$$x_{i1} = \begin{cases} 1, & \text{if } i \text{ is an element of } I_1, \\ 0, & \text{otherwise,} \end{cases}$$

and

$$x_{i2} = \begin{cases} 1, & \text{if } i \text{ is an element of } I_2, \\ 0, & \text{otherwise.} \end{cases}$$

Then we define a matrix $Q \in \mathbf{R}^{2n}$ by

$$\begin{array}{llll} q_{i1,i1} & = q_{i2,i2} & = \frac{1}{n} + r_i^2 M & \text{for all } i \in \{1,2,\dots,n\}, \\ q_{i1,i2} & = q_{i2,i1} & = 0 & \text{for all } i \in \{1,2,\dots,n\}, \\ q_{i1,j1} & = q_{i2,j2} & = M r_i r_j & \text{for all } i,j \in \{1,2,\dots,n\},\ i \neq j, \\ q_{i1,j2} & = q_{i2,j1} & = -M r_i r_j & \text{for all } i,j \in \{1,2,\dots,n\},\ i \neq j. \end{array}$$

Let us now consider the problem

$$(P_2') \qquad \begin{array}{ll} \min x^T Q x & \\ x_{i1} + x_{i2} = 1 & \text{for all } i = 1,2,\dots,n, \\ x_{i1}, x_{i2} \in \{0,1\} & \text{for all } i = 1,2,\dots,n. \end{array}$$

The constraints $x_{i1} + x_{i2} = 1$ imply that $x_{i1}x_{i2} = 0$ for all $i = 1, 2, \ldots, n$ so that x^TQx can be written as

$$\begin{aligned}
x^TQx &= \sum_{i=1}^{n}(x_{i1}^2 q_{i1,i1} + x_{i2}^2 q_{i2,i2}) + \sum_{i=1}^{n}\sum_{j=1;i\neq j}^{n} x_{i1}x_{j1}q_{i1j1} + \\
&\quad \sum_{i=1}^{n}\sum_{j=1;i\neq j}^{n} x_{i2}x_{j2}q_{i2j2} + \sum_{i=1}^{n}\sum_{j=1;i\neq j}^{n} (x_{i1}x_{j2}q_{i1j2} + x_{i2}x_{j1}q_{i2j1}) \\
&= \sum_{i=1}^{n}(\frac{1}{n} + Mr_i^2)(x_{i1}^2 + x_{i2}^2) + \sum_{i=1}^{n}\sum_{j=1;i\neq j}^{n} Mr_ir_jx_{i1}x_{j1} + \\
&\quad \sum_{i=1}^{n}\sum_{j=1;i\neq j}^{n} Mr_ir_jx_{i2}x_{j2} - 2M\sum_{i=1}^{n}\sum_{j=1;i\neq j}^{n} r_ir_jx_{i1}x_{j2} \\
&= \sum_{i=1}^{n}\frac{1}{n}(x_{i1}^2 + x_{i2}^2) + M\Big(\sum_{i=1}^{n} r_i^2x_{i1}^2 + \sum_{i=1}^{n}\sum_{j=1;i\neq j}^{n} r_ir_jx_{i1}x_{j1} + \\
&\quad \sum_{i=1}^{n} r_i^2x_{i2}^2 + \sum_{i=1}^{n}\sum_{j=1;i\neq j}^{n} r_ir_jx_{i2}x_{j2} - 2\sum_{i=1}^{n}\sum_{j=1;i\neq j}^{n} r_ir_jx_{i1}x_{j2}\Big) \\
&= \sum_{i=1}^{n}\frac{1}{n}(x_{i1}^2 + x_{i2}^2) + M(\sum_{i=1}^{n} x_{i1}r_i - \sum_{i=1}^{n} x_{i2}r_i)^2.
\end{aligned}$$

The constraints $x_{i1}, x_{i2} \in \{0, 1\}$ imply that $x_{i1}^2 = x_{i1}$ and $x_{i2}^2 = x_{i2}$ for all $i = 1, 2, \ldots, n$. Furthermore the conditions $x_{i1} + x_{i2} = 1$ for all $i = 1, 2, \ldots, n$ imply that

$$\sum_{i=1}^{n}\frac{1}{n}(x_{i1}^2 + x_{i2}^2) = 1.$$

Thus problem (P_2') can be written as:

$$\begin{aligned}
&\min\ 1 + M(\sum_{i=1}^{n} x_{i1}r_i - \sum_{i=1}^{n} x_{i2}r_i)^2 \\
&\quad x_{i1} + x_{i2} = 1 \text{ for all } i = 1, 2, \ldots, n, \\
&x_{i1}, x_{i2} \in \{0, 1\} \text{ for all } i = 1, 2, \ldots, n.
\end{aligned}$$

This is equivalent to

$$\begin{aligned}
&\min\ 1{+}M(\sum_{i\in I_1} r_i - \sum_{i\in I_2} r_i)^2 \\
&I_1 \cup I_2 = \{1, 2, \ldots, n\}, \\
&I_1 \cap I_2 = \emptyset, \\
&I_1, I_2 \subset \{1, 2, \ldots, n\}.
\end{aligned}$$

If there exists a 2-partition of the given integers $r_1, r_2, \ldots, r_n$ then there exists an optimal solution of problem (P_2') defined above with objective function value $c_{\text{opt}} = 1$.

Now apply algorithm A to (P_2'). If there exists a 2-partition, then A produces a feasible solution x_A with objective function value

$$c_A \leq (1+\varepsilon)c_{\text{opt}} = (1+\varepsilon) < M.$$

This and the definition of the objective function in (P_2') imply that

$$I_1 = \{i \in \{1,2,\ldots,n\} \mid x_{i1} = 1\}$$

and

$$I_2 = \{i \in \{1,2,\ldots,n\} \mid x_{i2} = 1\}$$

define a 2-partition of $r_1, r_2, \ldots, r_n$. Conversely, if there does not exist a 2-partition of $r_1, r_2, \ldots, r_n$ then $c_A \geq c_{\text{opt}} \geq 1 + M$.

Therefore we can use algorithm A to decide in polynomial time whether there exists a 2-partition of $r_1, r_2, \ldots, r_n$ which is impossible unless $\mathcal{P} = \mathcal{NP}$. □

(3.6) Corollary. *(P_2) and (P_2^+) are $\mathcal{NP}$-hard.*

Proof. Theorem (3.5) obviously implies the $\mathcal{NP}$-hardness of problem (P_2).

Let an instance of (P_2) be given and let $q = \min\{q_{ik,jl} \mid i,j \in \{1,2,\ldots,n\}, k,l \in \{1,2\}\}$.

If $q < 0$, we define a new matrix $Q' \in \mathbf{Q}^{2n \times 2n}$, $Q' \geq 0$, by setting $q'_{ik,jl} = q_{ik,jl} - q$ for all $i,j \in \{1,2,\ldots,n\}$ and $k,l \in \{1,2\}$.

Then

$$\begin{array}{ll} \min x^T Q' x & \\ x_{i1} + x_{i2} = 1 & \text{for all } i = 1,2,\ldots,n, \\ x_{i1},\ x_{i2} \in \{0,1\} & \text{for all } i = 1,2,\ldots,n \end{array}$$

is an instance of (P_2^+) with $Q' \geq 0$ and $x^T Q' x = x^T Q x - n^2 q$ for any feasible solution x. Since (P_2) is $\mathcal{NP}$-hard, the statement follows. □

4. The decomposition approach

The complexity of the placement problem suggests to decompose a large instance recursively into smaller ones, and this is indeed a common technique in electronic circuit layout. The most popular approach of the creation of a k-nary tree T, called the cut tree.

The root of T corresponds to all cells and the whole area of the circuit. For any tree node t, its k children $t_1, t_2, \ldots, t_k$ correspond to k rectangular subareas $a_{t_1}, a_{t_2}, \ldots, a_{t_k}$ of the rectangle a_t represented by t and k subsets $c_{t_1}, c_{t_2}, \ldots, c_{t_k}$ of the cell set c_t assigned to node

t. In addition, for each tree node t which is not a leaf it is specified how $a_{t_1}, a_{t_2}, \ldots, a_{t_k}$ must be combined to form the rectangular area a_t, see Figure 2 for an example with $k = 4$.

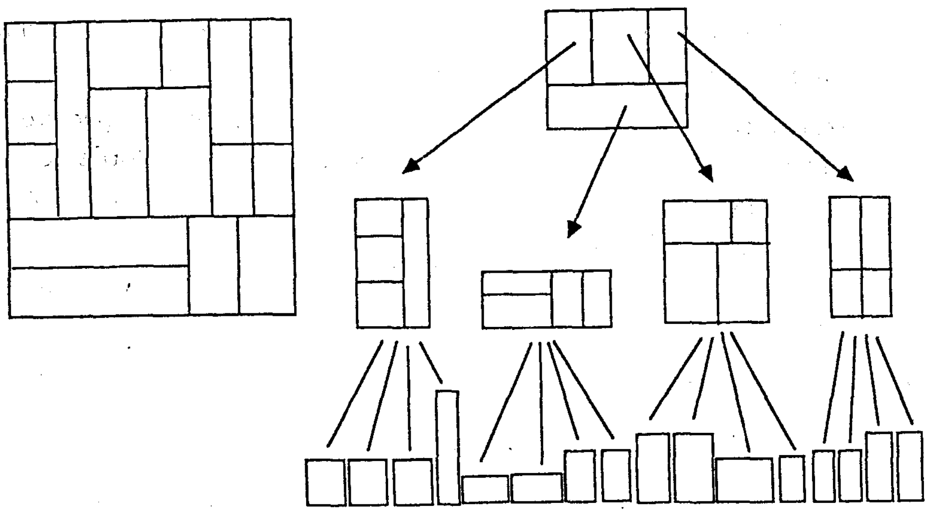

Figure 2

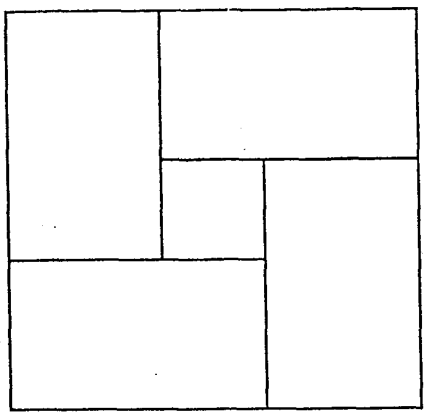

Figure 3

Such trees are usually constructed by bottom-up clustering, top down decomposition or a combination of both techniques (cf. MÜLLER (1990) for a detailed discussion). It is tried heuristically to make sure that little wiring is necessary between the sons of any tree node, respectively, the bulk of wiring is within the son nodes ("min-cut placement").

Although this approach has proved to be quite successful in practice, it has several drawbacks. One is that the complexity of the involved optimization problems grows considerably with k so that $k = 2$ is common practice whereas it would require $k \geq 5$ to obtain the layout displayed in Figure 3.

Another drawback is that global view is lost when working in a non-root node of the tree. This may lead to decisions on lower levels which are unfavorable in higher levels.

For the special case of the sea-of-cells layout style we consider here, we designed a decomposition scheme which overcomes such difficulties.

We denote by w respectively h the number of base cells in horizontal respectively vertical direction. Furthermore, we assume that the base cells are linearly ordered from 0 to $h \cdot w - 1$. For ease of exposition base cell k will also be denoted by the tupel (x, y), where $x = k \bmod w$, $y = \lfloor k/w \rfloor$ and therefore $k = y \cdot w + x$.

At the beginning of the decomposition scheme we consider the following 16 base cells $(\lfloor \frac{uw}{4} \rfloor, \lfloor \frac{vh}{4} \rfloor)$, $u = 0, 1, 2, 3$, $v = 0, 1, 2, 3$ and set $B^1 := \cup_{u=0}^{3} \cup_{v=0}^{3} \{(\lfloor \frac{uw}{4} \rfloor, \lfloor \frac{vh}{4} \rfloor)\}$.
Each base cell $b = (\lfloor \frac{uw}{4} \rfloor, \lfloor \frac{vh}{4} \rfloor) \in B^1$ represents an area of the master, i. e. a set of base cells, namely

$$\{(x,y) : \lfloor \frac{uw}{4} \rfloor \leq x < \lfloor \frac{(u+1)w}{4} \rfloor, \ \lfloor \frac{vh}{4} \rfloor \leq y < \lfloor \frac{(v+1)h}{4} \rfloor\}.$$

The set B^1 is called the set of representative base cells and an element $b \in B^1$ is called a **representative base cell** (for the first iteration).

For cell i to be placed we define a set $Z^1(i)$, $|Z^1(i)| \leq 16$, that consists of all feasible representative base cells for i. More precisely,

$$Z^1(i) = B^1 \setminus \{(b_x, b_y) \in B^1 \mid b_x + w_i > w \text{ or } b_y + h_i > h\}.$$

An instance of (P_{16}^{+}) is solved, where the variables $x_{ik} = 1$ are interpreted as cell i being assigned to the k-th variable of set $Z^1(i)$ $(i = 1, \ldots, n,\ k = 1, \ldots, |Z^1(i)| \leq 16)$. Thus, each cell is assigned to one of these 16 representative base cells.

In the second iteration each of the 16 areas of the master corresponding to the 16 elements of B^1 is divided into four equally-sized rectangular units. The lower left base cells of these units define the set B^2, i. e. the set of representative base cells in the next iteration. Put into a formula,

$$B^2 = \{(\lfloor \frac{uw}{8} \rfloor, \lfloor \frac{vh}{8} \rfloor), \ u = 0, 1, \ldots, 7, \ v = 0, 1, \ldots, 7\}.$$

Similarly we define sets $Z^2(.)$ which are the sets of feasible representative base cells for the cells in the second iteration.

Assume cell i was assigned to $b_0 = (\lfloor \frac{u_0 w}{4} \rfloor, \lfloor \frac{v_0 h}{4} \rfloor) \in B^1$ for $u_0, v_0 \in \{0, 1, 2, 3\}$ in the first iteration. Then, we determine $Z^2(i)$ such that

(1) $Z^2(i)$ consists of at most 16 elements of B^2.

(2) The distance (measured in the L_1-norm) beween b_0 and any element of $Z^2(i)$ must be less than or equal to the distance between b_0 and every element of $B^2 \setminus Z^2(i)$.

More precisely, we set

$$\begin{aligned} Z^2(i) = &\{(\lfloor \frac{uw}{8} \rfloor, \lfloor \frac{vh}{8} \rfloor) \mid \lfloor \frac{uw}{8} \rfloor + w_i \leq w,\ \lfloor \frac{vh}{8} \rfloor + h_i \leq h, \\ &u = \max\{0, 2u_0 - 1\}, 2u_0, \min\{w-1, 2u_0+1\}, \min\{w-1, 2u_0+2\}, \\ &v = \max\{0, 2v_0 - 1\}, 2v_0, \min\{h-1, 2v_0+1\}, \min\{h-1, 2v_0+2\}\}. \end{aligned}$$

Proceeding this way, the area represented by b_0 is covered by the area represented by $Z^2(i)$. Furthermore, if $|Z^2(i)| = 16$, the area represented by b_0 is symmetrically covered by the area represented by $Z^2(i)$. (Note that $Z^2(i)$ may represent an area which is larger than the area represented by b_0). Therefore, cell i is not restricted to the area represented by b_0, but can leave it in all directions. Thus, our decomposition scheme has the property that cells can move around.

The process of generating sets B^j, generating sets $Z^j(.)$ for all cells and solving an instance of (P_{16}^+) is continued, until in the final iteration j_0 the set B^{j_0} consists of all base cells of the master.

We strongly believe that our hierarchical decomposition approach is one of the key reasons for the practical success of our placement method.

Figure 4 shows an example of these refinement stages. Assume that in step 1 a certain cell is assigned to the marked base cell in the first picture. Then the sixteen equally marked base cells in the second picture are the feasible base cells for the cell in step 2. Assuming that in step 2 the cell is assigned to the differently marked base cell, the sixteen feasible base cells in the final step 3 are marked accordingly in the third picture.

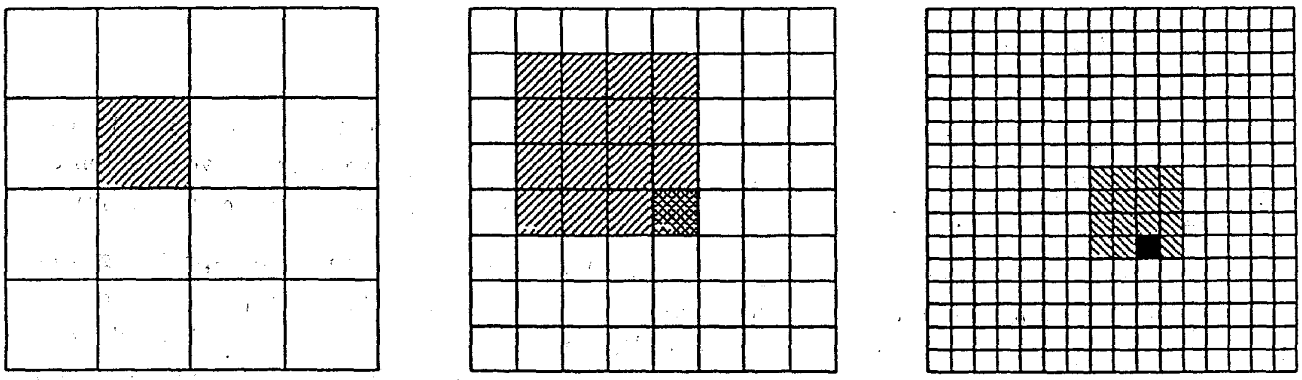

Figure 4

Working in this scheme, our problem reduces to solving a sequence of instances of (P_{16}^{+}):

$$\begin{aligned}
&\min x^T Q x \\
&\sum_{k=1}^{|Z^j(i)|} x_{ik} = 1 \quad \text{for all } i = 1,2,\dots,n, \\
&x_{ik} \in \{0,1\} \quad \text{for all } i = 1,2,\dots,n, \text{ and for all } k = 1,2,\dots,|Z^j(i)| \le 16.
\end{aligned}$$

The variables $x_{ik} = 1$ can be interpreted as cell i being assigned to the k-th element of set $Z^j(i)$ $(i = 1,\dots,n,\ k = 1,\dots,|Z^1(i)| \le 16)$. Let $z(ik)$ denote the base cell corresponding to variable x_{ik}. The coefficients of the matrix $q_{ik,jl}$ are determined by the formula

$$\begin{aligned}
q_{ik,jl} &= c_{ij} \cdot \operatorname{dist}(i, z(ik), j, z(jl)) + \lambda \cdot o(i, z(ik), j, z(jl)), \\
q_{ik,ik} &= \sum_{r \in R} c_{ir} \operatorname{dist}(i, z(ik), r, p(r)).
\end{aligned}$$

Recall that $c_{ij} \geq 0$ denotes the affinity coefficient between cells i and j. The number $\operatorname{dist}(i, z(ik), j, z(jl))$ is the Manhattan distance between cells i and j when assigned to base cells $z(ik)$ and $z(jl)$, respectively. The number of overlapping units is $o(i, z(ik), j, z(jl)) \geq 0$, if cells i and j are assigned to base cells $z(ik)$ and $z(jl)$ respectively. R is the set of pad cells and $p(r)$ denotes the (predefined) location of pad cell r. Using this matrix Q with an appropriate penalty parameter λ, solutions of (P_{16}^{+}) with an equal distribution of cells to representative base cells are preferred in all iterations j, where $|B^j| < n$. This is due to the fact that the more cells are assigned to the same base cell, the more pairs of cells have positive overlap coefficients. In an iteration j with $|B^j| > n$ we strive for a placement without overlaps.

5. Graphtheoretic formulation

In the following we want to present a graphtheoretic formulation of our placement model. Based on this new formulation we will develop heuristics for an approximative solution of the problem. For ease of exposition let us first assume that the set of pad cells is empty.

To an instance of the placement problem (P_m) we associate a complete n-partite graph $G_c = (V_c, E_c, w_c)$ as follows. Every node $u \in V_c$ corresponds to a unique variable x_{ik}, $i = 1,\dots,n$, $k = 1,\dots,m$ and vice versa. Between two nodes $u, v \in V_c$, $u \neq v$, where u and v correspond to the variables x_{ik} and x_{jl} respectively, we introduce an edge e if and only if $i \neq j$. The weight $w_c(e) = w_c(u,v)$ is given by the coefficient $q_{ik,jl}$ of the matrix Q. In the following we will use the notation ik for the node representing the variable x_{ik}.

Figure 5 shows the graph G_c for $n = 4$, $m = 3$.

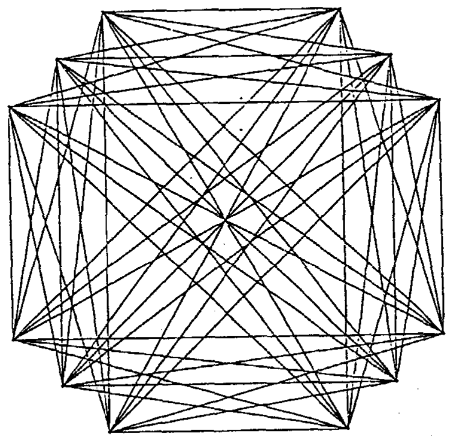

Figure 5

A **stable set** in G_c is a subset of V_c in which no two nodes are connected by an edge. A **clique** is a subset of nodes such that every pair of nodes is connected by an edge.

By construction, all maximal stable of G_c are maximum, and the maximum stable sets are precisely the sets $S_i = \{ik \mid k = 1, 2, \ldots, m\}$ for $i = 1, \ldots, n$. Furthermore, any feasible solution of (P_m) is in 1-1-correspondence to a clique in G_c of cardinality n, and its objective function value is the total weight of this clique.

Thus, the placement problem is equivalent to the following graphtheoretic problem.

"Find among all cliques of cardinality n in $G_c = (V_c, E_c, w_c)$ one of minimum weight."

We assumed that the set of pad cells R is empty, i. e. $q_{ik,ik} = 0$ for all $i = 1, \ldots, n$, $k = 1, \ldots, m$. This assumption can be made without loss of generality, since otherwise a matrix Q' can be defined via

$$\begin{aligned}
&q'_{ikjl} = q_{ikjl} + \frac{1}{n-1} q_{ikik} + \frac{1}{n-1} q_{jljl} \\
&\text{for all } i, j = 1, \ldots, n,\ i \neq j,\ k, l = 1, \ldots, m, \\
&q'_{ikil} = q_{ikil} \text{ for all } i = 1, \ldots, n,\ k, l = 1, \ldots, m,\ k \neq l, \\
&q'_{ikik} = 0 \text{ for all } i = 1, \ldots, n,\ k = 1, \ldots, m.
\end{aligned}$$

Obviously, every feasible solution of (P) with matrix Q corresponds to a feasible solution of (P) with matrix Q' such that the objective function values are equal, and vice versa. Thus, we will asssume in the following that $q_{ikik} = 0$ for all $i = 1, \ldots, n$, $k = 1, \ldots, m$.

In the remainder of this section we will present the heuristics we designed and implemented for a solution of this graphtheoretic formulation of the placement problem (P_{16}^{+}).

For every $v \in V_c$ let $S(v)$ denote the unique maximal stable set S_i such that $v \in S_i$.

The first heuristic is a greedy type heuristic. We start with a clique of cardinality 2. In each step, we add a node to the present clique until maximum cardinality is reached.

Heuristic 1

Input: $G_c = (V_c, E_c, w_c)$.

Output: A clique Q in G_c of cardinality n.

```
(1) Let $w_c(u_0, v_0) = \min\{w_c(u,v) \mid (u,v) \in E_c\}$.
    Set $Q = \{u_0, v_0\}$ and $\overline{Q} = S(u_0) \cup S(v_0)$.
(2) As long as $|Q| < n$ perform the following steps.
  (2.1) Let $\sum_{v \in Q} w_c(v, w_0) = \min\{\sum_{v \in Q} w_c(v, w) \mid w \in V_c \setminus \overline{Q}\}$.
  (2.2) Set $Q = Q \cup \{w_0\}$ and $\overline{Q} = \overline{Q} \cup S(w_0)$.
```

It is easy to see that the worst case running time of this algorithm is $O(mn^3)$. In each iteration the heuristic adds a node w_0 to the present clique and all nodes belonging to the maximal stable set $S(w_0)$ except w_0 are deleted simultaneously.

The second heuristic idea avoids the simultaneous deletion of more than one node. We iteratively delete nodes until a clique of cardinality n is left. To this end we assign weights to the nodes in G_c and to the maximum stable sets in G_c. In each step we determine a maximum stable set S_0 with the highest weight and a node u_0 of S_0 with highest weight (among all nodes of S_0). The node u_0 is deleted from G_c. These steps are repeated until every maximum stable set consists of only one node. These remaining nodes define a clique of maximum cardinality.

Let $\delta(v)$ denote the set of edges of G_c incident to $v \in V_c$.

Heuristic 2

Input: $G_c = (V_c, E_c, w_c)$.

Output: A clique Q in G_c of cardinality n.

```
(1) For each $v \in V_c$ compute $c(v) = \frac{1}{|\delta(v)|} \sum_{(u,v) \in E_c} w_c(u,v)$.
(2) For each maximum stable set $S$ in $G_c$ let $\gamma(S) = \max_{v \in S} c(v) / \min_{v \in S} c(v)$. Set
    $Q = \emptyset$.
(3) As long as $|V_c| > n$ perform the following steps.
```

```
    (3.1) Compute $S_0$ with $\gamma(S_0) = \max\{\gamma(S) \mid S$ is a maximum stable set in $G_c, |S| \geq 2\}$. Let $v_0$ be a node in $S_0$ with $c(v_0) = \max_{v \in S_0} c(v)$.
    (3.2) Set $V_c = V_c \setminus \{v_0\}$ and $E_c = E_c \setminus \delta(v_0)$.
    (3.3) Update the values $c(v)$ for all $v \in V_c$ and compute $\gamma(S)$ for all maximum stable sets $S$ in $G_c$.
(4) Set $Q = V_c$.
```

It is obvious that the worst case running time of this algorithm is $O(m^2n^2)$.

The two heuristics presented above can be applied at any stage of the decomposition approach. The third heuristic is developed for the solution of instances of (P_{16}^+) after the first iteration has been finished. Suppose we have performed an iteration. For each cell i let o_i be the total number of overlapping basecells between cell i and all other cells. The location cell i is assigned to is represented by a node ik in the new graph G_c of the next iteration. We use o_i as the weight of the maximum stable set $S(ik)$ and sort these maximum stable sets according to decreasing weights. Without loss of generality let $S_1, S_2, \ldots, S_n$ denote this ordered sequence. We start with m cliques of cardinality 1 corresponding to the m nodes of the maximum stable set S_1 and extend each of these cliques by one node in every step. Therefore, we will end up with m cliques of cardinality n. Each of these m cliques is finally tried to be improved by applying a one-exchange heuristic (i. e. a node of the present clique is replaced by another one, if this substitution reduces the objective function value of the old solution). Finally we choose a clique with lowest objective function value..

Heuristic 3

Input: $G_c = (V_c, E_c, w_c)$.

Output: A clique Q of cardinality n.

```
(1) Let $\{v_i^1, v_i^2, \ldots, v_i^m\}$ be the nodes of $S_i$. Set $Q_1^k = \{v_1^k\}$, and $q(Q_1^k) = 0$ for all $k = 1, \ldots, m$. ($Q_i^k$ denotes the $k$th clique in step $i$.) Set $i = 2$.
(2) As long as $i \leq n$ perform the following steps.
    (2.1) Set $I = \{1, \ldots, m\}$.
    (2.2) For all $k = 1, \ldots, m$
        (2.2.1) Compute
                (*) $q_i^k = q(Q_{i-1}^k) + \min_{l \in I}( \sum_{u \in Q_{i-1}^k} w_c((u, v_i^l)))$.
```

(2.2.2) Let $l_0 \in I$ be an index such that $q_i^k = q(Q_{i-1}^k) + \sum_{u \in Q_{i-1}^k} w_c((u, v_i^{l_0}))$ and set $q(Q_i^k) = q_i^k$.

(2.2.3) Set $I = I \setminus \{l_0\}, Q_i^k = Q_{i-1}^k \cup \{v_i^{l_0}\}$

(2.3) Set $i = i + 1$.

(3) Set $Q_k = Q_n^k$ for all $k = 1, \ldots, m$.

(4) For each of the cliques Q_k, $k = 1, 2, \ldots, m$, perform 1-exchanges as long as $q(Q_k)$ can be decreased.

(5) Determine the clique Q with $q(Q) = \min\{q(Q_k) \mid k = 1, 2, \ldots, m\}$.

The running time of this algorithm depends on the number of 1-exchanges examined in step (4). If the running time spent in step (4) is bounded by $O(m^2n^2)$ then we obtain an overall worst case running time for heuristic 3 of $O(m^2n^2)$.

Step (2.2) to determine the cliques of the i-th iteration could be substituted by solving a matching problem between the nodes of the maximum stable set S_i and the m cliques of the $(i-1)$-th iteration.

We have performed several tests to evaluate the heuristics outlined above (for details, see WEISMANTEL (1992)). To this end we generated instances (P_{16}^+) that come up during the decomposition approach of the placement problem. Summing up, it may be said that heuristic 2 provides the best solutions. However, the running times necessary to find a solution of the problem instances by heuristic 2 are much higher than those of the other algorithms. Nevertheless, we decided to solve the instances of (P_{16}^+) at all stages of the decomposition approach using heuristic 2, since our main goal was to find good solutions.

6. Implementation details

In this section we discuss some details that make our approach work in practice. We present several ideas that help to reduce the huge demand on memory space and running time. For demonstration purposes we use an industry example with 1021 logic cells, 1023 nets and 63 pad cells.

A first problem is to store all relevant information as compactly as possible, but without loosing too much time for retrieving this information from the data structures. We would get the fastest access to the data if we could store Q as a full matrix. In our example, this would require 540 megabytes of memory space for $m = 16$ if we assume that one floating point/integer number needs 4 bytes of memory space. To reduce this memory demand we analyze the objective function more carefully. The entries of Q are given as

$$q_{ik,jl} = c_{ij} \cdot d_{ik,jl} + \lambda \cdot o_{ik,jl},$$

where

$$c_{ij} = \text{affinity coefficient between cells } i \text{ and } j,$$
$$d_{ik,jl} = \text{Manhattan distance between cell } i \text{ assigned to base cell } k \text{ and cell } j \text{ assigned to base cell } l,$$
$$o_{ik,jl} = \text{number of overlapping base cell units if cells } i \text{ and } j \text{ are assigned to base cells } k \text{ and } l, \text{ respectively.}$$

First, consider the coefficients c_{ij} for $i, j = 1, \ldots, n$. Each cell has relatively few pins. Thus, each cell is only connected to a small number of nets. In addition, nets connecting many cells very probably have to be routed over the whole chip. Such large nets give little information on where to place the cells. Therefore, we eliminate large nets. What a large net is, depends on the features of a given chip. We analyse the structure of the nets and then decide heuristically which nets are called large. In our example, nets with more than 33 cells are eliminated and each cell is only connected to 1% of the cells on the average. This way one can obtain an enormous reduction on the number of nonzero coefficients c_{ij}.

Second, consider the coefficients $d_{ik,jl}$. In our model, we use the Manhattan metric as a distance measure and assume that a coordinate system is underlying with its origin at the lower left corner of the master. In the sea-of-cells layout style each cell i is of a predefined type $t(i)$. Two types are different if they differ in their width, height or the list of pins. (In our example there are only 34 different types of logic cells.) If we want to calculate the distance between cells i and j when assigned to base cells k and l, respectively, it is sufficient to compute the distance between the two types $t(i)$ and $t(j)$ when a cell of type $t(i)$ is assigned to base cell k and a cell of type $t(j)$ is assigned to base cell l. Furthermore, we use the fact that the distance in Manhattan metric is the sum over the distances in horizontal and vertical directions, and that the distance is invariant with respect to translation. The latter fact implies that we can assume without loss of generality that cell i is located at the origin.

Taking all these observations into account, the coefficients $d_{ik,jl}$ can be calculated by

$$d_{ik,jl} = \operatorname{dist}\Big(t(i), \tbinom{0}{0}, t(j), \tbinom{|l_x - k_x|}{0}\Big) + \operatorname{dist}\Big(t(i), \tbinom{0}{0}, t(j), \tbinom{0}{|l_y - k_y|}\Big),$$

where p_x and p_y denote the horizontal and vertical coordinates, respectively, of location p.

Analogous arguments apply for the third type of coefficients $o_{ik,jl}$ except that we have to multiply the number of overlapping units in x-direction by those in y-direction. So, the coefficients $o_{ik,jl}$ can be calculated by

$$o_{ik,jl} = \mathrm{ov}_x\Big(t(i), \tbinom{0}{0}, t(j), \tbinom{|l_x - k_x|}{0}\Big) \cdot \mathrm{ov}_y\Big(t(i), \tbinom{0}{0}, t(j), \tbinom{0}{|l_y - k_y|}\Big),$$

where ov_x and ov_y are the overlaps in horizontal and vertical direction, respectively (see Figure 6). This way the number of different coefficients that have to be stored can be reduced enormously. In our example we just need 2.8 megabytes of memory space instead of 540 megabytes. Another advantage of this decomposition of the matrix Q is that the relevant values have to be computed only once.

Still, computation of exact overlappings is time consuming. In the first iterations the number of node pairs ik and jl with positive overlap coefficient $o_{ik,jl}$ is quadratic in n. Because of the few feasible locations many cells must overlap. Therefore we decided to consider a simplified evaluation of the overlap term. For each base cell we compute the number of nodes jl with the property that when cell i is assigned to location l, it covers this base cell. For a base cell l, let $z(l)$ denote this number. Based on the information provided by the values $z(l)$ we developed improvement heuristics. For example, we try to get a better distribution of the cells over the chip or to get rid of overcapacitated locations (these are locations l such that the values $z(l)$ exceed a certain number) still taking our estimate of the wiring length exactly into account. Or vice versa, we try to improve our estimate of the wiring length (maintaining a reasonable distribution of the cells).

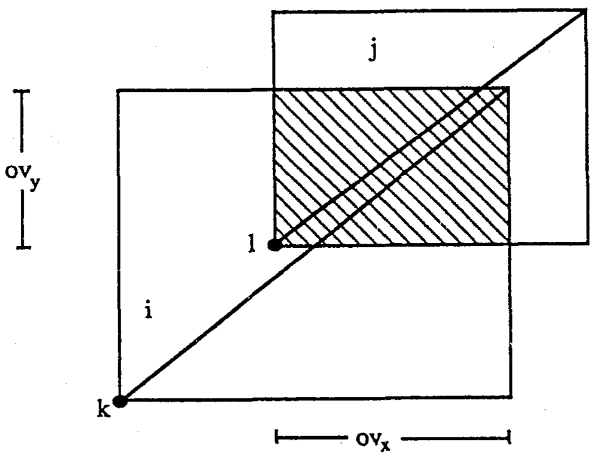

Figure 6

In later iterations the situation changes. The number of node pairs ik and jl with positive overlap coefficients $o_{ik,jl}$ reduces considerably. Now the problem is to get for each node ik fast access to those nodes having positive overlap coefficient with ik. We have designed suitable data structures that allow for an efficient computation of exact overlaps.

Combining the ideas indicated in this section we were able to reduce the running time considerably.

7. Computational results

Our experimental software has been compared to industrially used placement algorithms. The placements obtained were assessed in two ways. A first measure for the quality of a placement is the so-called estimated wiring length which is computed by adding up the lengths of the minimum spanning trees of all nets. This is only an estimate since nets are routed using Steiner trees and since in order to make the routing feasible, detours have often to be taken. Nevertheless, this is an accepted number for comparing placements. In addition, we employed a maze router used in industry to really compute routings for our placements. This maze router was used as a black box, i. e., we could not tune the router with respect to our placement approach. For a real chip, this router can almost never completely route a nonoverlapping placement of the cells. The interesting measure here is the number of two-pin connections necessary to complete the wiring produced by the maze router. This corresponds to the amount of work that has to be spent interactively to find a feasible wiring. Our experience with this router has shown that the placement routine influences the number of non-connected pin pairs only to a certain extent. A lot of non-connected pin pairs are caused by the greedy type approach of the router, i. e. pin pairs are connected one at a time so that connections between succeeding pin pairs may be blocked. Nevertheless, this number is an indication for the routability of a given placement so that the estimated wiring length in combination with the number of missing connections allows for a reasonable comparison of different placement approaches.

We report here on experiments with three test circuits. Circuit 1 consists of 1022 cells and 1025 nets, its density is 50%. Circuit 2 consists of 2293 cells and 2776 nets, its density is 23.6%. Circuit 3 consists of 2670 cells and 3128 nets, its density is 50.3%. The **density** of a circuit is the total area of the cells to be placed divided by the area of the master. Table 1 presents the results of a min-cut based placement procedure (cf. LAUTHER (1979) for a detailed discussion of the procedure), Table 2 shows the results for Gordian, a method based on an energy model (KLEINHANS, SIGL, JOHANNES (1988)). Column 2 in these tables contains the estimated wiring length, column 3 gives the number of missing connections for circuits 1, 2 and 3, respectively. The running times in column 4 were obtained on 8 and 16 MIPS workstations, respectively.

Tables 3, 4 and 5 present the results obtained by the experimental version of our method. Column 2 in Tables 3, 4 and 5 shows the estimated wiring length for circuits 1, 2 and 3, respectively. The number of missing connections for circuits 1, 2 and 3 are shown in column 3 of Tables 3, 4 and 5, respectively.

In placement routines an estimate of the total wiring length is minimized, whereas the routability condition is almost neglected. To overcome this difficulty a lot of sophisticated little tricks exploiting technical information must be implemented. To this end years of testing are necessary (consider, for example, the development of Gordian in JOHANNES, JUST, ANTREICH (1983), JUST, KLEINHANS (1985), KLEINHANS, SIGL, JOHANNES (1988), KLEINHANS, SIGL, JOHANNES, ANTREICH (1991)).

Up to now we only use a quite simple approach to reflect the wireability condition in our placement algorithm. Namely, we extend certain cells by at most one unit in horizontal and vertical direction. We bound the total amount of extension by a constant factor c,

$0 \leq c \leq 1$, i. e. the total amount of extension is $c \cdot (1 - \text{density}) \cdot b$, where b denotes the number of base cells of the master.

Furthermore, we used Heuristic 2 followed by an improvement algorithm based on one-exchanges for a solution of the instances of P_{16}^{+} (cf. sections 5 and 6). Tables 3, 4 and 5 present the results we obtained for circuit 1, circuit 2 and circuit 3, respectively. Row 1 in Tables 3, 4 and 5 shows the results, if the total amount of extension is bounded by $c = 20\%$. In rows 2 and 3 of these tables we evaluate the final placement of the 3 circuits, if c is set to 40% and 60%, respectively. Column 4 of the tables shows the running times of our implementation obtained on a 27 MIPS computer.

Finally, Figures 7, 8 and 9 show the final placements for the three circuits resulting from the version of our method displayed in row 2 of Tables 3, 4 and 5.

The results show that the solutions provided by the Min-Cut placement procedure are significantly worse than the ones provided by Gordian and our method. On the average, the number of non-connected pin pairs in the placements obtained by applying Gordian or our method are approximatively equal.

However, in nearly all examples the estimated wiring lengths of our solutions are better than those of Min-Cut and Gordian. In case of circuit 2 the estimated wiring length of the solution provided by our algorithm ($c = 20\%$) is 16% better that Gordian's solution. Tables 3, 4 and 5 show that the higher the total amount of extension is chosen, the fewer pin pairs are not connected and the higher the estimated wiring length is. The right choice of this parameter depends on special features of a circuit. For circuit 1, the choice $c = 40\%$ seems to be best, whereas for circuits 2 and 3 the best results are obtained, when c is set to 20% and 60%, respectively. Based on these results we propose that for circuits with a very homogeneous cell- and net structure the factor c should be chosen small ($c = 20\%$). For circuits with a more heterogenous cell or net structure a higher value should be assigned to c ($c = 40\%$). If, additionally, the density of the circuit exceeds 50%, a choice of $c = 60\%$ seems to be reasonable.

In terms of running time we are far away from comparison to any industrially used algorithm. However, if one realizes that it lasts from 10 hours (circuit 1) up to two days (circuit 3) for routing a given placement on a workstation (using the industry software that was available to us), the running time of our placement procedure is still acceptable. Taking into account that the implementation of our approach is a prototype, in which technical information is not fully exploited (see the extension factor c) and that the running time was not optimized, the results indicate that it is worthwile to further explore our approach.

circuit	estimated wiring length	number of non-connected pin pairs	CPU-Time (min./ 8 mips)
circuit 1	194732	24	4:25
circuit 2	796622	33	15:54
circuit 3	623159	551	21:04

Table 1 (Min Cut)

circuit	estimated wiring length	number of non-connected pin pairs	CPU-Time (min./ 16 mips)
circuit 1	189683	19	3:43
circuit 2	652129	51	3:13
circuit 3	506160	253	6:52

Table 2 (Gordian)

options	estimated wiring length	number of non-connected pin pairs	CPU-Time (min./ 27 mips)
$c = 20\%$	180101	17	26:07
$c = 40\%$	185592	14	26:08
$c = 60\%$	191577	14	26:09

Table 3 (circuit 1)

options	estimated wiring length	number of non-connected pin pairs	CPU-Time (min./ 27 mips)
$c = 20\%$	553575	48	129:39
$c = 40\%$	570387	57	129:24
$c = 60\%$	576845	50	129:37

Table 4 (circuit 2)

options	estimated wiring length	number of non-connected pin pairs	CPU-Time (min./ 27 mips)
$c = 20\%$	472473	309	159:22
$c = 40\%$	486175	278	159:21
$c = 60\%$	497285	260	159:23

Table 5 (circuit 3)

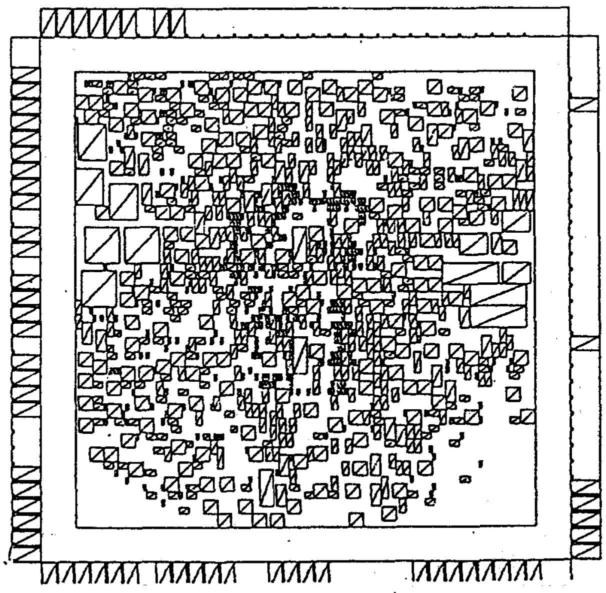

Figure 7 (circuit 1)

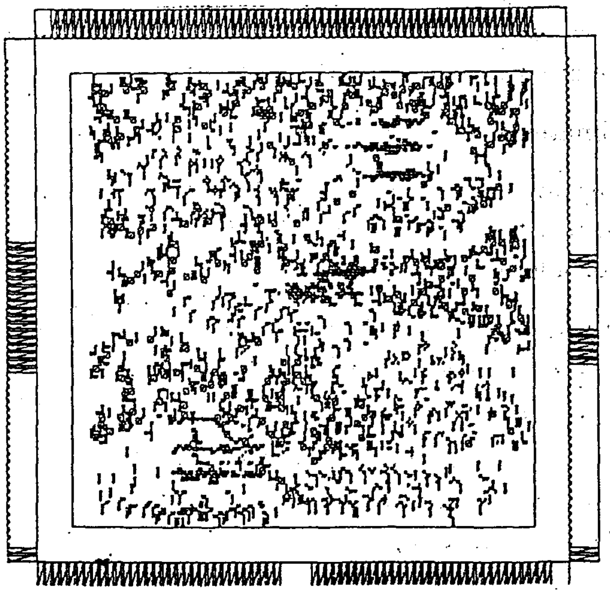

Figure 8 (circuit 2)

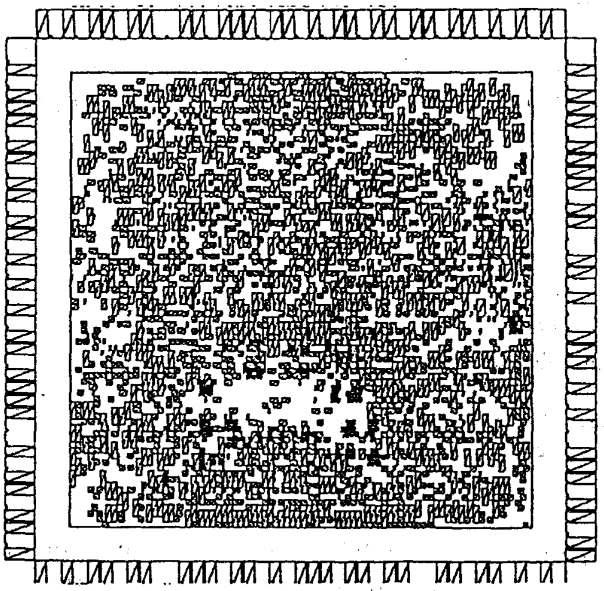

Figure 9 (circuit 3)

8. Final Remarks

We have presented a new method for the placement of electronic circuits in the sea-of-cells layout style and we have discussed the results of an experimental version of our implementation. There are still several possibilities for further improvement.

(1) Exact pin coordinates are not taken into account yet.

(2) As an initial step for our placement routine, clustering could be performed (i. e. several cells are put together and treated as a super cell). This would speed up computations considerably.

(3) We could modify our decomposition approach by solving a sequence of instances of (P_9^+) instead of a sequence of instances of (P_{16}^+). This would further reduce the running time, but might have a negative effect on the quality of the placements.

(4) Up to now we only use a simple approach to reflect the wireability condition in our placement algorithm (cf. Section 7). Improvements should be possible if we take technical details of the cells and of the master into account.

Our approach is not limited to the layout style discussed in this paper. It can also be applied if pad cells are not preassigned (methods like Gordian are based on an energy model and cannot handle this case) and it can be extended to the case that cells are positioned on different layers.

Acknowledgements

This work was financially supported by Siemens AG München. Special thanks are due to R. Dachauer and Dr. M. Bartholomeus for providing real-world data and for performing the benchmarks with Gordian and Min-Cut.

References

Garey, M. R. & Johnson, D. S. (1979), "Computers and Intractability. A Guide to the Theory of $\mathcal{NP}$-Completeness", Freeman, New York, 1979.

Johannes, F. M., Just, K. M. & Antreich, K. J. (1983), "On the force placement of logic arrays", Proc. 6-th., European Conference on Circuit Theory and Design, pp. 203–206, 1983.

Just, K. M. & Kleinhans, J. M. (1985), "On the simultaneous placement of modules of integrated circuits", AEÜ, Vol. 39, No. 4, pp. 217–224, 1985.

Karp, R.M. (1972), "Reducibility among combinatorial problems", Plenum Press, New York, 1972.

Kleinhans, J. M., Sigl, G. & Johannes, F. M. (1988), "Gordian: A new global optimization/ rectangle dissection method for cell placement", IEEE Int. Conference on CAD ICCAD-88, pp. 506–509, 1988.

Kleinhans, J. M., Sigl, G. & Johannes, F. M., Antreich, K.J. (1991), "Gordian: VLSI Placement by Quadratic Programming and Slicing Optimization", IEEE Transactions on computer aided design,Vol. 10, No. 3, 1991.

Lauther, U. (1979), "A min-cut placement algorithm for general cell assemblies based on a graph representation", ACM/ICEE Proc. 16th DAC, 1–10,1979.

Lengauer, T. (1990), "Combinatorial Algorithms for Integrated Circuit Layout", Wiley-Teubner, New-York, 1990.

Müller, R. (1990), "Hierarchisches Floorplanning mit integrierter globaler Verdrahtung", Dissertation, Universität GH Paderborn, 1990.

Weismantel, R. (1992), "Plazieren von Zellen: Theorie und Lösung eines quadratischen 0/1 Optimierungsproblems", Dissertation, Technische Universität Berlin, 1992.