GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization

Jürgen M. Kleinhans, Georg Sigl, Frank M. Johannes, and Kurt J. Antreich, *Senior Member, IEEE*

***Abstract*—In this paper we present a new placement method for cell-based layout styles. It is composed of alternating and interacting global optimization and partitioning steps that are followed by an optimization of the area utilizaiton. Methods using the divide-and-conquer paradigm usually lose the global view by generating smaller and smaller subproblems. In contrast, GORDIAN maintains the simultaneous treatment of all cells over all global optimization steps, thereby considering constraints that reflect the current dissection of the circuit. The global optimizations are performed by solving quadratic programming problems that possess unique global minima. Improved partitioning schemes for the stepwise refinement of the placement are introduced. The area utilization is optimized by an exhaustive slicing procedure. The placement method has been applied to real world problems and excellent results in terms of both placement quality and computation time have been obtained.**

I. Introduction

THE ACCEPTANCE of cell-based design styles is considerably influenced by the quality and the speed of the available design tools. In this paper we present strategies and algorithms of a new placement tool named GORDIAN, which has been successfully applied to all cell-based layout styles and particularly to large circuits.

Cell-based design is performed with predefined or adaptable functional units—cells which are taken from a well tested cell library. The most common layout styles are row-oriented standard cells and gate arrays. Standard cell circuits may either be complete chips or may form macros (building blocks) of hierarchical macrocell designs. The new sea-of-gates layout style exhibits features of the traditional gate array and macrocell concepts. As with macrocell designs, sea-of-gates cells can also vary considerably in size and aspect ratio. However, a sea-of-gates circuit can consist of thousands or tens of thousands of cells. The large circuit size and the variability of the cells, combined with the fixed area and routing resources of the master, makes the layout synthesis of sea-of-gates circuits very difficult.

The task of placement, the first step in the physical design process, is to calculate the positions of the cells. Since the quality of the placement determines the minimal achievable area and wiring length of a circuit, it has a large impact on production yield and circuit performance. Good placement tools, therefore, have to meet high requirements: they have to enable the successful completion of routing within minimal or given area and must be able to deal with large designs.

The difficulty of the placement problem increases as the cell count grows. Therefore, the classical approach to VLSI placement is based on the divide-and-conquer paradigm. Important representatives of this approach are based on min-cut graph partitioning (e.g., [1]–[4]). However, min-cut algorithms like those of Kernighan and Lin [5] and Fiduccia and Mattheyses [6] are iterative improvement heuristics that depend on an initial partition. Ng *et al.* [7] pointed out that it might be necessary to select one partition computed from many randomly generated starting partitions to obtain a good solution. They proposed a clustering algorithm that constructs a contracted network to be partitioned by the min-cut algorithm and in this way obtained improved results. Suaris and Kedem [4] extended the Fiduccia–Mattheyses bisection algorithm to quadrisection and reported improved results when applied to standard cell placement.

Recently, alternative algorithms that model the placement problem as a linear or nonlinear continuous optimization problem have been studied. In contrast to the min-cut approach, geometric information about cell and chip dimensions and pin locations can be used directly. Usually no starting solution is needed and all modules (cells) are treated simultaneously. Among these approaches are methods using physical (force or electrical network) analogies [8]–[12] and eigenvector methods [13]–[15]. Some of these methods apply partitioning to recursively create smaller subproblems. However, they restrict the simultaneous optimization to the initial step.

Getting stuck at local optima is a major drawback of partitioning-based methods. Efforts have been made to deal with this problem, especially to improve the widely used min-cut procedure—e.g., terminal propagation has been introduced by Lauther [1] and Dunlop and Kernighan [2] to consider the nets that connect cells in different regions. This global connectivity problem also arises with continuous optimization-based methods when applied to smaller and smaller subproblems.

The placement method GORDIAN [16] presented here has the unique feature of maintaining simultaneity over *all* optimization steps. The acronym GORDIAN stands for the two main parts of the method: global optimization and rectangle dissection, which is based on improved partitioning schemes.

With GORDIAN, the placement problem is formulated as a sequence of quadratic programming problems derived from the entire connectivity information of the circuit. An increasing number of constraints restricting the freedom of movement of the modules is imposed, reflecting the results of successively refined partitionings. In this way, on each level of refinement, a global placement of the modules is obtained simultaneously for all subproblems, avoiding any dependence on a processing sequence. The application of GORDIAN to standard cell and macrocell benchmarks from [17] has been discussed in [18].

Manuscript received November 9, 1989. This paper was recommended by Associate Editor R. H. J. M. Otten.

J. M. Kleinhans is with the Siemens Corporate Research and Development, Applied Computer Science and Software, Systems Design Automation, D-8000 Munich 83, Germany.

G. Sigl, F. M. Johannes, and K. J. Antreich are with the Institute of Computer-Aided Design, Department of Electrical Engineering, Technical University of Munich, D-8000 Munich 2, Germany.

IEEE Log Number 9040959.

The extension of the procedure to the sea-of-gates layout style was presented in [19].

In the following sections, a detailed description of the components of the method is given and further results are presented. In Section II, the procedure is outlined. The quadratic programming approach to global placement is described in Section III. Section IV discusses fundamental and improved partitioning schemes. Section V explains how the final placement is obtained in accordance with the specific layout style. Space and time complexity is discussed in Section VI. Results for various standard cell and sea-of-gates circuits with up to over 6000 cells are discussed in Section VII.

II. Outline of the Procedure

The placement procedure GORDIAN is composed of alternating and interacting global optimization and partitioning steps that are followed by a final placement step that adapts the global placement to style-dependent constraints. The data flow between these main steps is illustrated in Fig. 1.

The input to GORDIAN consists of a net list, an extract of the cell library, and a description of the geometry of the chip. The net list can be written as a binary relation $\mathfrak{I} \subseteq \mathfrak{N} \times \mathfrak{M}$, where $\mathfrak{N}$ and $\mathfrak{M}$ are the index sets of the nets and the modules, respectively. A connection of net ν to module μ is represented by $(\nu, \mu) \in \mathfrak{I}$; the set of modules connected by net ν is $\mathfrak{M}_\nu = \{ \mu \in \mathfrak{M} \mid (\nu, \mu) \in \mathfrak{I} \}$. The dimensions (width and height) of each rectangular module as well as the locations of its pins are taken from the cell library. For sea-of-gates circuits, the description of the chip geometry includes the basic cell array dimensions defining the possible module locations on the master. With standard cell designs the number of rows to be used must be given. For macrocell circuits an estimated placement area has to be described. The positions of the pad cells are needed independently of the layout style.

The main loop of GORDIAN is formed by an iteration of global optimization and partitioning steps. They aim at minimal wirelength and at a uniform distribution of the modules over the available placement area.

The global optimization starts with an initial (root) region that comprises the whole core area of the chip and contains all modules to be placed. One constraint fixes the center of gravity of all these modules to the center of this region. In each partitioning step the module set is further divided and the placement regions are dissected into son regions accordingly, thereby establishing new constraints for the next global optimization step. The partitioning generates a slicing tree [20], [21] whose nodes correspond to the regions containing subsets of the modules.

This loop of global optimization and partitioning steps (Fig. 1) is repeated until each region contains at most k modules, where k is a predefined constant. For standard cell circuits the modules are finally gathered into rows. For macrocell and sea-of-gates circuits, the possible slicing dissections are enumerated. The allocation of the modules to the leaf regions is derived from their global placement, thereby avoiding a costly permutation. This allows the method to be applied to regions containing $k = 30$ or more modules even with large sea-of-gates designs. The result of this exhaustive slicing optimization is a shape function for each of these regions. It consists of area minimal rectangles circumscribing all enumerated module allocations with different aspect ratios. Finally, these shape functions are simultaneously evaluated to produce a placement of the modules that globally optimizes the area utilization.

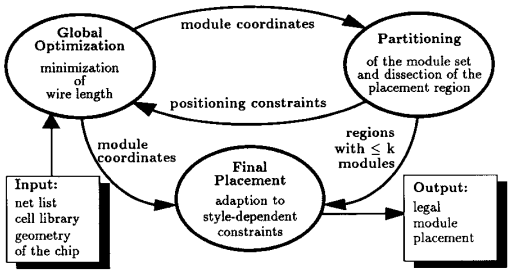


Fig. 1. Data flow in the placement procedure GORDIAN.

III. Global Placement by Quadratic Programming

In each global optimization step, a quadratic programming problem is derived from the circuit connectivity (the net list) and from the dissection of the placement area on the respective level of partitioning. The solution of this quadratic programming problem is a *global placement* of the modules.

3.1. Problem Formulation

The objective function of the global optimization step is based on the rubber band lengths of the nets. The length L_ν of a net ν is measured by the sum of the squared distances from its pins to the nets center coordinates (x_ν, y_ν)

$$L_\nu = \sum_{\mu \in \mathfrak{M}_\nu} \left[(x_\mu + \xi_{\nu\mu} - x_\nu)^2 + (y_\mu + \eta_{\nu\mu} - y_\nu)^2 \right] \quad (1)$$

where $(\xi_{\nu\mu}, \eta_{\nu\mu})$ are the coordinates of a pin connected to net ν relative to the center coordinates (x_μ, y_μ) of its module μ (in Fig. 2 the sum of the squared lengths $l_{\nu\mu}$ of the dashed lines is the length L_ν of net ν).

To each net ν an individual weight $w_\nu \geq 1$ is assigned. A high net weight groups modules that are connected by this possibly critical net closer together. Thus the objective function is the weighted sum of the squared rubber band lengths of the nets:

$$\Phi = \tfrac{1}{2} \sum_{\nu \in \mathfrak{N}} L_\nu \cdot w_\nu. \quad (2)$$

In order to reduce the number of variables we substitute the coordinates of the nets by the mean values of the coordinates of their pins. This way, the net variables are eliminated. Due to the net model chosen (Fig. 2) this is equivalent to replacing each net by all two-point connections of its pins (a clique). The objective function, which now depends only on the module coordinates, can be written in matrix form where the constant terms are deleted:

$$\Phi(\boldsymbol{x}, \boldsymbol{y}) = \tfrac{1}{2} \boldsymbol{x}^T \boldsymbol{C} \boldsymbol{x} + \boldsymbol{d}_x^T \boldsymbol{x} + \tfrac{1}{2} \boldsymbol{y}^T \boldsymbol{C} \boldsymbol{y} + \boldsymbol{d}_y^T \boldsymbol{y}. \quad (3)$$

The vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ denote the coordinates of the m movable modules $\mu \in \mathfrak{M}_m \subset \mathfrak{M}$ in the m-dimensional vector space $\mathfrak{R}^m$. The system matrix $\boldsymbol{C}$ and the vectors $\boldsymbol{d}_x$ and $\boldsymbol{d}_y$ are set up according to the procedure *set_up_objective_function* shown in Fig. 3. For each net ν, the edges of the clique which replaces the net, are weighted by the value e. It is set to $e = 2/p$; $p = |\mathfrak{M}_\nu|$ (disregarding the net weight w_ν for the moment) since this way, the total edge weight of the clique amounts to $(2/p) \cdot (p \cdot (p-1)/2) = p - 1$, which is the number of edges in a spanning tree that connects all pins of net ν.

The matrix $\boldsymbol{C}$ is positive definite if all movable modules are connected to fixed modules (e.g., pad cells) either directly or indirectly. This condition holds for all useful net lists, since

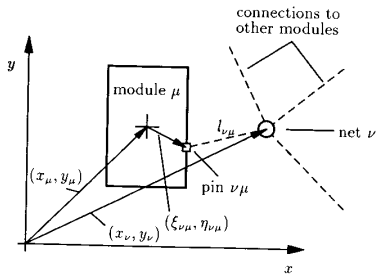


Fig. 2. Net modeling and coordinates of modules and nets.

```
procedure set_up_objective_function
C := 0; d_x := 0; d_y := 0;
for ν ∈ 𝒩
  e := w_ν · 2/|𝓜_ν|;
  for μ ∈ 𝓜_ν ∩ 𝓜_m
    c_μμ += e · (|𝓜_ν| − 1);
    d_x,μ += ξ_νμ · e · (|𝓜_ν| − 1);
    d_y,μ += η_νμ · e · (|𝓜_ν| − 1);
    for λ ∈ 𝓜_ν \ {μ}
      if λ ∈ 𝓜_m
        c_μλ −= e;
        d_x,μ −= ξ_νλ · e;  d_y,μ −= η_νλ · e;
      else /* λ ∈ 𝓜_f */
        d_x,μ −= e · (x_λ + ξ_νλ);
        d_y,μ −= e · (y_λ + η_νλ);
      endif
    endfor
  endfor
endfor
endprocedure
```

Fig. 3. Setting up the objective function for global placement.

each module should be accessible from the outside of the circuit. The vectors $\boldsymbol{d}_x$ and $\boldsymbol{d}_y$ originate from the contributions of the fixed modules $\lambda \in \mathfrak{M}_f = \mathfrak{M} \setminus \mathfrak{M}_m$ with coordinates (x_λ, y_λ) and the pin coordinates of all modules.

Since (3) is separable into $\Phi(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x}) + \varphi(\boldsymbol{y})$, we restrict our discussion to the part of the objective function that depends on the x-coordinates

$$\phi(\boldsymbol{x}) = \tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{C}\boldsymbol{x} + \boldsymbol{d}^T\boldsymbol{x} \tag{4}$$

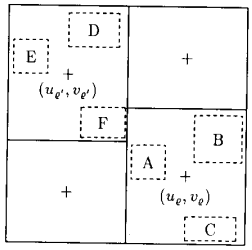
where $\boldsymbol{d} = \boldsymbol{d}_x$. The matrix $\boldsymbol{C}$ is the same for both objective functions $\varphi(\boldsymbol{y})$ and $\phi(\boldsymbol{x})$. They differ only in the vector $\boldsymbol{d}$ due to the different x- and y-coordinates of the pins and the fixed modules.

At the top optimization level ($l = 0$), all m modules to be placed belong to the root region which covers the whole placement area available to the modules. At the lth level of optimization, the placement area is divided into $q \le 2^l$ regions $\rho \in \mathfrak{R}^{(l)}$, each containing a subset $\mathfrak{M}_\rho \subseteq \mathfrak{M}_m$ of modules, where $\mathfrak{R}^{(l)}$ is the index set of the regions on level l. The centers (u_ρ, v_ρ) of these regions impose constraints on the global placement of the modules:

$$\boldsymbol{A}^{(l)}\boldsymbol{x} = \boldsymbol{u}^{(l)} \tag{5}$$

such that the area weighted mean value of the coordinates of modules $\mu \in \mathfrak{M}_\rho$, i.e., the center of gravity, corresponds to the center of region ρ. The entries $a_{\rho\mu}$ of the $\langle q \times m \rangle$-matrix $\boldsymbol{A}^{(l)}$ depend on which module (occupying F_μ units of area) belongs to which region ρ:

$$a_{\rho\mu} = \begin{cases} F_\mu \Big/ \sum_{\mu \in \mathfrak{M}_\rho} F_\mu, & \text{if } \mu \in \mathfrak{M}_\rho \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$



$$\boldsymbol{A}^{(l)} = \begin{array}{c} \\ \vdots \\ \varrho \\ \varrho' \\ \vdots \end{array} \begin{array}{c} \begin{array}{cccccccc} A & B & C & D & E & F & \cdots \end{array} \\ \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ * & * & * & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & * & * & * & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \end{array}$$

Fig. 4. The constraints for global placement.

Fig. 4 illustrates how the constraint matrix $\boldsymbol{A}^{(l)}$ is set up for modules belonging to two different regions ρ and ρ'. Each column of $\boldsymbol{A}^{(l)}$ contains just one nonzero entry depicted by "*" in the row corresponding to the region the module belongs to.

Combining the objective function (4) and the constraints (5), the following *linearly constrained quadratic programming problem* (LQP) is obtained:

$$\text{LQP: } \min_{\boldsymbol{x} \in \mathfrak{R}^m} \left\{ \phi(\boldsymbol{x}) = \tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{C}\boldsymbol{x} + \boldsymbol{d}^T\boldsymbol{x} \,\middle|\, \boldsymbol{A}^{(l)}\boldsymbol{x} = \boldsymbol{u}^{(l)} \right\}. \tag{7}$$

Since $\phi(\boldsymbol{x})$ is a convex function ($\boldsymbol{C}$ is positive definite) and the linear equality constraints (5) define a convex subspace of $\mathfrak{R}^m$, (7) has a unique global minimum $\phi(\boldsymbol{x}^*)$.

This particular modeling results in a LQP which is based on the entire circuit connectivity information at each level of optimization. The model at level l is derived from the model at level $l-1$ by refining the constraints. Thus the placement problem is mapped to a sequence of optimally solvable problems LQP.

Other placement methods that iteratively alternate global optimization and partitioning steps like [10] and [12] differ considerably from GORDIAN in the way they treat the regions on a level of refinement: for each region they solve a separate optimization problem regarding modules that belong to other regions as fixed. Thus their solution depends on the sequence in which the regions are considered.

3.2. Solution Method

The q linear equality constraints restrict the freedom of movement of the modules to a $(m-q)$-dimensional subspace of $\mathfrak{R}^m$. Visually, one module of each subset $\mathfrak{M}_\rho$ has to be moved such that the center of gravity constraint imposed on the modules in this region is satisfied, while all other modules are free to move anywhere. This means that the m-dimensional coordinate vector $\boldsymbol{x}$ can be partitioned into $m-q$ independent variables $\boldsymbol{x}_i$ and q dependent variables $\boldsymbol{x}_d$:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_{d\langle q \rangle} \\ \boldsymbol{x}_{i\langle m-q \rangle} \end{bmatrix}. \tag{8}$$

This ordering of the coordinates corresponds to a partitioning of the constraint matrix $\boldsymbol{A}$ into:

$$\boldsymbol{A}_{\langle q \times m \rangle} = [\boldsymbol{D}_{\langle q \times q \rangle}\, \boldsymbol{B}_{\langle q \times m-q \rangle}]. \tag{9}$$

In the matrix A, there is exactly one entry for each module in each column (cf. (6) and Fig. 4). Therefore, D can be chosen to be a diagonal matrix made of nonzero entries of A taking, for numerical reasons, the biggest entry of each row of A. The dependent variables x_d and the vector x now can be expressed as a function of the independent variables x_i:

$$x_d = -D^{-1}Bx_i + D^{-1}u \tag{10}$$

$$x = Zx_i + x_0 \tag{11}$$

with

$$Z = \begin{bmatrix} -D^{-1}B \\ I \end{bmatrix} \quad \text{and} \quad x_0 = \begin{bmatrix} D^{-1}u \\ 0 \end{bmatrix}. \tag{12}$$

For the vector x_0 any choice is appropriate that satisfies $Ax_0 = u$, e.g., the modules can initially be put onto the centers of their regions.

Substituting (11) into (4), a $(m - q)$-dimensional *unconstrained quadratic programming problem* (UQP) in the variables x_i is obtained

$$\text{UQP:} \min_{x_i \in \Re^{m-q}} \left\{ \psi(x_i) = \tfrac{1}{2} x_i^T Z^T C Z x_i + c^T x_i \right\} \tag{13}$$

with $c^T = (Cx_0 + d)^T Z$.

Since C is positive definite and the columns of Z form a basis of a $(m - q)$-dimensional subspace of $\Re^m$, the matrix $Z^T CZ$ is also positive definite. Thus to determine the global minimum solution x^* of (7), simply means to solve the equation system

$$Z^T C Z x_i^* = -c \tag{14}$$

derived from (13) by setting the gradient $\nabla\psi(x_i)$ to zero, and to substitute x_i^* into $x^* = Zx_i^* + x_0$.

While C is a sparse matrix, $Z^T CZ$ is usually dense, so it is essential not to require $Z^T CZ$ explicitly when solving (14). Therefore, direct solvers and iterative methods which need $Z^T CZ$ are impracticable. However, a well-suited iterative solution method for this class of problems is the *conjugate-gradient method* [22]–[24]. This method computes the solution using only products of the matrix $Z^T CZ$ with a vector, and does not explicitly require the matrix elements. Using appropriate data structures for C and Z, only sparse matrix-vector products have to be performed, resulting in an efficient solution procedure.

IV. Improved Partitioning Schemes

During partitioning the module set and the placement area are recursively divided. Constraints are imposed on module subsets to get a better distribution of the modules over the whole placement area. GORDIAN does not use the partitioning principle to reduce the problem size, but to restrict the freedom of movement of the modules. Since these restrictions influence the following global optimizations and eventually fix approximate positions of the modules very close to their final placement, the decisions in the partitioning step are crucial. The partitioning decision is derived mainly from global placement. However, it should also be based on the number of nets crossing the new cut line. This number can be minimized by variation of the cut position, as described in Section IV-4.1, or by exchanging modules between the new subsets created, as explained in Section IV-4.2. Furthermore, the partitioning decisions can be improved by verifying them as often and as early as possible and to correct them if necessary. This can be achieved by repartitioning, which is described in Section IV-4.3.

The partitioning step divides each region $\rho \in \Re^{(l)}$ into two son regions $\rho', \rho'' \in \Re^{(l+1)}$. The module set $\mathfrak{M}_\rho$ is bipartitioned into the subsets $\mathfrak{M}_{\rho'}$ and $\mathfrak{M}_{\rho''}$. The sums of the module areas of both subsets determine the dissection of the rectangular area of region ρ. The area $F_{\rho'}$ of region ρ' is defined by

$$\frac{F_{\rho'}}{F_\rho} = \frac{\sum_{\mu' \in \mathfrak{M}_{\rho'}} F_{\mu'}}{\sum_{\mu \in \mathfrak{M}_\rho} F_\mu} = \alpha \tag{15}$$

where α is the desired area ratio. The area $F_{\rho''}$ is given by $F_\rho - F_{\rho'}$. In the case of a vertical cut the modules $\mu \in \mathfrak{M}_\rho$ of region ρ are sorted by their global placement coordinates x_μ and are assigned to ρ' and ρ'' such that

$$\forall_{\mu' \in \mathfrak{M}_{\rho'}} x_{\mu'} \leq \min_{\mu'' \in \mathfrak{M}_{\rho''}} \{x_{\mu''}\}. \tag{16}$$

The most obvious way of partitioning is to predefine $\alpha \approx 0.5$ and to alternate the direction of the cut on each level. This leads to regions with approximately the same area and aspect ratio.

The quality of a partition can be measured by the cut value $c_\rho(\alpha)$ which is the sum of the weights of nets that cross the cut line

$$c_\rho(\alpha) = \sum_{\nu \in \mathfrak{N}_c} w_\nu$$

with

$$\mathfrak{N}_c = \left\{ \nu \in \mathfrak{N} \,\middle|\, \mathfrak{M}_\nu \cap \mathfrak{M}_{\rho'} \neq \varnothing \wedge \mathfrak{M}_\nu \cap \mathfrak{M}_{\rho''} \neq \varnothing \right\}. \tag{17}$$

If the partitioning of a region is determined according to (15) and (16), the cut values have not yet been taken into account. Therefore, GORDIAN applies improved partitioning schemes to cut the Gordian knot. These try to minimize the influence of the partitioning step on the final layout by taking advantage of the global placement. In the following sections, three different methods for improving the partitioning are presented.

4.1. Improved Partitioning by Variation of Cut Direction and Position

To avoid large cut values the position of the cut can be varied. Going from left to right through the list of modules $\mathfrak{M}_\rho$ sorted by their global placement coordinates, and drawing a vertical cut line after each module, $c_\rho(\alpha)$ may be determined for all values of α. Fig. 5 illustrates this analysis for the example of Fig. 6. The value of α should usually be around 0.5. Thus it is selected within the range $|1 - 2\alpha| \leq \gamma \leq 1$ where $c_\rho(\alpha)$ is minimum. Experiments indicate that the parameter γ should not exceed the value of 0.3, since for larger values the dimensions of regions may differ too much, resulting in wasted area.

For each region ρ the cut values $c_\rho(\alpha)$ are calculated for both vertical and horizontal cuts. The lower value in the specified range of α suggests the cut direction. Should this choice create son regions with extreme aspect ratios, the cut is then made in the other direction using the proper value of α.

4.2. Improved Partitioning by Module Interchange

Another method to reduce cut values is to interchange modules between the subsets $\mathfrak{M}_{\rho'}$ and $\mathfrak{M}_{\rho''}$ of the initial partition of a region ρ derived from global placement according to (15) and

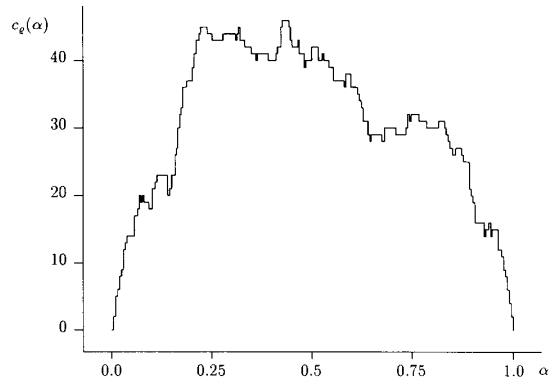


Fig. 5. Cut function $c_\varrho(\alpha)$.

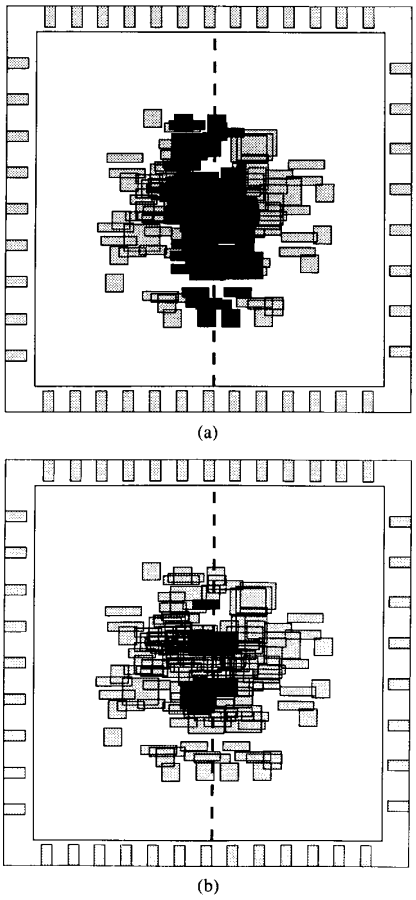


Fig. 6. Combination of global placement and min-cut ($\gamma = 0.5$). (a) Min-cut initial partition. (b) Modules interchanged by min-cut.

(16). This is the well-known min-cut approach [5]. In GORDIAN, the min-cut algorithm of Fiduccia and Mattheyses [6] is optionally applied. Since initial partitions which have been derived from global placement are improved, the global connectivity is considered. Furthermore, terminal propagation [1], [2] is used. Since a global placement is at hand, more detailed information about the positions of the modules can be used than with usual min-cut, where terminal propagation has to be derived from the center coordinates of the regions the modules belong to.

Based on the global placement, min-cut is supplied with four subsets of modules for each region ρ. $\mathfrak{M}_\rho$ is divided into $\mathfrak{A}_L$, $\mathfrak{A}_F$, $\mathfrak{B}_F$, and $\mathfrak{B}_L$, such that for a vertical cut $x_{\mu\mathfrak{A}_L} \leq x_{\mu\mathfrak{A}_F} \leq x_{\mu\mathfrak{B}_F} \leq x_{\mu\mathfrak{B}_L}$. Fig. 6(a) shows this initial partition of the module set of a macrocell circuit, where min-cut is applied after the first global optimization step. The modules belonging to the sets $\mathfrak{A}_F$ and $\mathfrak{B}_F$ are highlighted by darker shading. Only the modules belonging to these two sets are free to move, whereas the modules $\mu \in \mathfrak{A}_L \cup \mathfrak{B}_L$ are locked. The size of the subsets can be controlled by the parameter γ:

$$\sum_{\mu\in\mathfrak{A}_F} F_\mu + \sum_{\mu\in\mathfrak{B}_F} F_\mu \simeq \gamma \cdot \sum_{\mu\in\mathfrak{M}_\rho} F_\mu \wedge \sum_{\mu\in\mathfrak{A}_F} F_\mu \simeq \sum_{\mu\in\mathfrak{B}_F} F_\mu. \tag{18}$$

A value of $\gamma = 1$ means that all modules of $\mathfrak{M}_\rho$ are treated by min-cut, a smaller value restricts the min-cut algorithm to modules close to the cut line—in Fig. 6(a), $\gamma = 0.5$ is chosen.

Min-cut converts the sets $\mathfrak{A}_F$ and $\mathfrak{B}_F$ into the sets $\mathfrak{A}'_F$ and $\mathfrak{B}'_F$, minimizing the cut value. New son regions ρ' and ρ'' are created with the modified module sets $\mathfrak{A}'_F$ and $\mathfrak{B}'_F$ such that $\mathfrak{M}_{\rho'} = \mathfrak{A}_L \cup \mathfrak{A}'_F$ and $\mathfrak{M}_{\rho''} = \mathfrak{B}_L \cup \mathfrak{B}'_F$. Fig. 6(b) indicates that only the few highlighted modules will actually be interchanged. Obviously the initial partition derived from global placement will be modified only near the cut depicted by the dashed line.

4.3. Repartitioning

During the first global optimization steps, modules may be clustered around the centers of their regions. If these regions are cut close to the center, the assignment of a module to one of the son regions may be fairly arbitrary, since many modules have approximately the same coordinates. The quality of the partitioning of a region $\rho \in \mathfrak{R}^{(l-1)}$ can be valued after the global optimization on level l. A large overlap of the global placement of the module subsets belonging to the son regions $\rho', \rho'' \in \mathfrak{R}^{(l)}$ indicates a bad partitioning since many modules of region ρ' tend to migrate to region ρ'' and vice versa.

Fig. 7 describes repartitioning which follows the global optimization step in the main loop of the GORDIAN procedure. If an overlap of the module subsets $\mathfrak{M}_{\rho'}$ and $\mathfrak{M}_{\rho''}$ is detected, i.e., for a vertical cut if

$$\exists_{\mu'\in\mathfrak{M}_{\rho'}} x_{\mu'} > \min_{\mu''\in\mathfrak{M}_{\rho''}} \{x_{\mu''}\} \tag{19}$$

for all pairs of son regions $\rho', \rho'' \in \mathfrak{R}^{(l)}$ derived from a common father region $\rho \in \mathfrak{R}^{(l-1)}$, then the module sets $\mathfrak{M}_{\rho'}$ and $\mathfrak{M}_{\rho''}$ are merged in $\mathfrak{M}_\rho = \mathfrak{M}_{\rho'} \cup \mathfrak{M}_{\rho''}$ and repartitioned according to (15) and (16). The repeated global optimization works with the same number of constraints as the previous one, but with an improved module to region assignment. This new global placement usually shows reduced or eliminated overlap of the module subsets. Experiments indicate that one repartitioning step suffices to largely reduce the overlap.

V. Final Placement

The result of the alternating global optimization and partitioning steps is a global placement and a slicing structure with regions containing k or less modules. Since this placement contains overlapping modules and has to be adapted to a specific design style, a final placement step has to follow. In a standard-

```
procedure GORDIAN
  l := 1;
  global_optimize(l);
  while (∃_ϱ |M_ϱ| > k)
    for each ϱ ∈ R^(l)
      partition(ϱ, ϱ', ϱ'');
    endfor
    l := l + 1;
    setup_constraints(l);
    global_optimize(l);
    repartition(l);
  endwhile
  final_placement();
endprocedure

procedure repartition (l)
  if overlap exists
    for each ϱ ∈ R^(l-1)
      merge_regions(ϱ, ϱ', ϱ'');
      partition(ϱ, ϱ', ϱ'');
    endfor
    setup_constraints(l);
    global_optimize(l);
  endif
endprocedure
```

Fig. 7. GORDIAN with repartitioning procedure.

cell design the modules are collected in rows, for macrocell and sea-of-gates circuits an optimization of the area utilization is performed, packing the modules in a compact slicing structure.

5.1. Standard Cell Final Placement

In standard cell designs the modules are of approximately the same height but sometimes of fairly differing widths. The chip area is determined by the widths of the channels between the cell rows and by the lengths of the rows including feedthroughs for nets crossing the rows. The goal is to obtain narrow channels with equally distributed low wiring density and rows with equal length.

In GORDIAN, the final placement for standard cells proceeds similarly to the method proposed by Dunlop and Kernighan [2]. To collect the modules into r horizontal rows, they are sorted by their y-coordinates and divided into r subsets by $r - 1$ horizontal cuts, such that $y_{\mu_1} \leq \cdots \leq y_{\mu_i} \leq \cdots \leq y_{\mu_r}$, where module μ_i belongs to the ith row, numbered from bottom to top. The sequence of the modules within the rows is determined by their x-coordinates.

With this allocation procedure, which tries to change the global placement as little as possible, narrow channels and low wirelength can be achieved. To ensure equal row lengths, the number of feedthroughs is estimated. Rows with a large number of feedthroughs are made shorter than the average row length and vice versa. The row lengths are varied within a 1–5% deviation from the average row length. To achieve the desired row length as exactly as possible, modules with y-coordinates close to the cut line are exchanged between neighboring rows if necessary.

5.2. Macrocell and Sea-of-Gates Final Placement

When the alternating global optimization and partitioning steps are completed, regions have been created that contain k or less modules. For these regions, an *exhaustive slicing optimization* (ESO) is performed which generates an optimal slicing structure in accordance to the global placement of the modules. Otten [13] published a heuristic to determine an optimized slicing structure for overlapping modules. Recently, van Ginneken [25] presented a polynomial algorithm to derive all possible slicing dissections of small sets of modules from a global placement.

The number $s(k)$ of different slicing dissections of a rectangle into k subrectangles is shown in Table I. Min-cut or clustering algorithms, which have no neighboring information from module coordinates, have to choose one assignment of k modules to k regions from $k!$ permutations. Since there are as much as $p(k) = k! \cdot s(k)$ possible placements, the enumeration of slicing structures is usually limited to $k \approx 5$.

However, in GORDIAN, module coordinates which are available from wirelength minimization in the global optimization step provide a criterion for module allocation. Therefore, the algorithm of van Ginneken [25] is applied to all ESO regions. It allows to enumerate all possible slicing dissections for module subsets with up to $k = 35$ modules even for large sea-of-gates circuits.

Fig. 8 shows the ESO procedure of GORDIAN. The procedure starts with the enumeration of all area minimal placements for each region that contains k or less modules. These placements are represented by a *shape function* [3], [26]–[28] for each ESO region. All area minimal placements of the whole circuit are obtained by recursively computing the shape function of the root of the slicing tree. After the selection of an appropriate root shape, a top-down traversal of the slicing tree that fixes the final placement is performed. Additionally it chooses the shape of each module. A module may possess more than one shape if it can be rotated or if there are different cell templates available from the library. By this ESO procedure, GORDIAN performs a global area optimization since all enumerations are evaluated simultaneously.

Fig. 9 shows a typical result of this exhaustive slicing optimization process. It depicts the root shape functions of a sea-of-gates circuit with over 6000 modules (see Section VII) for different values of the enumeration parameter k. Each point corresponds to the upper right corner of the circumscribing rectangle of an overlap-free placement. The fixed dimensions of the sea-of-gates master and the boundary curve $h = (\Sigma_\mu F_\mu / w)$ restrict the region of feasible placements to the shaded area in Fig. 9. The influence of the parameter k on the shown shape functions is obvious. A higher value of k results in lower area and more shapes within the feasible placement area. With a value of $k \leq 3$ no feasible placement can be achieved because of the bad area utilization. With growing k the area utilization increases and the shape function gets closer and closer to the boundary hyperbola.

However, it is not the best idea to make k as large as possible. Experiments with GORDIAN, when applied to designs with a large number of modules, indicate that k should be just as low as needed for a good area utilization, since for higher values of k the quality of the placement in terms of wirelength usually becomes worse due to the earlier termination of the global optimization and partitioning loop.

VI. Complexity of the Method

Space complexity: There is one system matrix $\boldsymbol{C}$ for both the x- and y-coordinates and for all global optimization steps. It is stored in a list structure with $O(m + N + P)$ memory space, where m, N, P are the numbers of movable modules, nets and pins, respectively. For larger circuits, P and N grow proportionally to m. The constraint matrix $\boldsymbol{A}$ that changes on every level, can be stored in a vector of length m (cf. (6)). The slicing tree has $2m - 1$ nodes. Thus the space complexity of the global optimization and the partitioning steps is $O(m)$.

Time complexity: Each iteration step in the global optimization takes time proportional to $(m + N + P)$, which is $O(m)$. The number n_l of iterations needed to solve each of the quadratic placement problems (7) depends on how tight the bounds on the accuracy of the solution are set. A practical limit for n_l is a value proportional to $m^{0.5}$. The partitioning of q regions based on sorting the modules takes time proportional to $q \cdot$

TABLE I
NUMBER OF SLICING DISSECTIONS AND PLACEMENTS

k	1	2	3	4	5	6	7	8	9	10
$s(k)$	1	2	6	22	90	394	1806	8558	41 586	206 098
$p(k)$	1	4	36	528	10 800	283 680	$9.1 \cdot 10^6$	$3.5 \cdot 10^8$	$1.5 \cdot 10^{10}$	$7.5 \cdot 10^{11}$

```
procedure ESO
  for all regions ϱ with |M_ϱ| ≤ k modules
    Determine the shape function of region ϱ by
    enumeration of all slicing structures
    that can be derived from the global
    placement coordinates of the modules;
  endfor
  Recursively compute the shape function of the root region
  bottom up from the shape functions of all ESO regions;
  Select one shape for the root region;
  Traverse the slicing tree top down
  to determine the module coordinates and shapes;
endprocedure
```

Fig. 8. Exhaustive slicing optimization procedure.

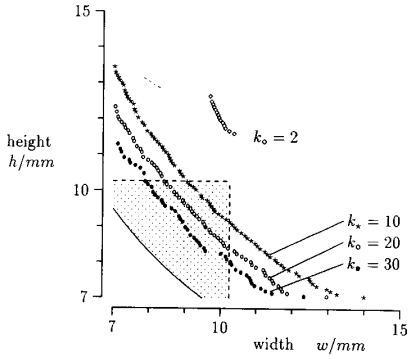


Fig. 9. Shape functions of circuit *sog6*.

$(m/q) \cdot \log(m/q)$ which is $O(m \cdot \log m)$. A balanced slicing tree has $\log m$ levels. Thus the total time complexity of global optimization and partitioning is $O(m^{1.5} \cdot \log^2 m)$.

The space and time complexity of the exhaustive slicing optimization mainly depends on how high the enumeration parameter k has been chosen and on the number of different shapes the modules possess. In [25] the time complexity of the slicing enumeration algorithm is shown to be $O(k^6)$. However, since k is fixed during the placement of a circuit, the time complexity of the ESO procedure grows linearly with m.

VII. EXPERIMENTAL RESULTS

GORDIAN has been implemented in the C language and is running on workstations and main frames. To investigate the efficiency of the GORDIAN placement procedure, it was applied to standard cell blocks of hierarchical designs, as well as to whole standard cell and sea-of-gates circuits. The different partitioning schemes presented in Section IV were compared for sea-of-gates circuits [19] and standard cell circuits. In most cases the synergy of global optimization and min-cut (Section IV-4.2) worked best.

TABLE II
CHARACTERISTICS OF STANDARD CELL BLOCKS AND CIRCUITS

Circuit	#cells	#pads	#nets	#pins	#rows
scb1	182	58	290	1140	4
scb2	324	470	737	2306	27
scb3	310	486	1094	3598	23
scb4	589	425	1165	4719	13
scb5	837	614	1333	4605	8
scb6	923	595	1379	5088	8
scb7	867	516	1507	6069	15
scb8	2158	957	3664	14 242	40
scb9	2481	1109	4265	16 995	43
*Primary1**	752	81	904	2941	15
*Struct**	1888	64	1920	5471	18
*Primary2**	2907	107	3029	11 226	24
*Biomed**	6417	97	5742	21 040	37

7.1. Standard Cell Circuits

Table II summarizes the characteristics of standard cell blocks and circuits which have been treated. The circuits marked by an asterisk are benchmarks distributed for the 1990 International Workshop on Layout Synthesis at MCNC [29]. For the standard cell blocks *scb1* to *scb9*, the pads column depicts the number of connectors (boundary pins).

Table III compares the results yielded by GORDIAN to those obtained from other tools, one based on min-cut, the other on simulated annealing [30]. The results are compared in terms of block area after final routing. In Table III bold numbers indicate the best results. The blocks were routed by the global and final routers of the VENUS CAD system [31]. For small circuits the simulated annealing tool gives the best results. However, for blocks with more than 1000 cells and nets, GORDIAN performs better. The gap between annealing and GORDIAN becomes larger with increasing circuit size. The annealing performance is worse for the large circuits because CPU-time becomes too expensive and faster cooling schedules lead to suboptimal results. The CPU-times given in Table III are measured on a 15 MIPS main frame computer.

Fig. 10 shows a plot of a design where the area of the standard cell blocks dominates the chip size. The chip represents a processing unit out of a series of chips for a main frame computer [32]. It contains 33 600 equivalent gate functions in the two large standard-cell blocks, 4.5-kb memory in 8 RAM cells, and one block with hard macros. The chip area has been remarkably reduced by putting all standard cells into just two blocks and by applying the placement procedure GORDIAN to these standard cell blocks (cf., the entries *scb8* and *scb9* in Table III). Several other circuits with comparable complexity have been successfully designed with the VENUS system including GORDIAN.

Tables IV–VII compare GORDIAN against the min-cut based placement method of the VENUS CAD system for the benchmark circuits from [29]. The comparisons are performed in terms of circuit area after completed wiring, the wiring length

TABLE III
COMPARISON OF RESULTS FOR STANDARD CELL BLOCKS

Circuit	Area After Routing/mm²		
	GORDIAN	Min-Cut	Annealing
scb1	2.7	3.1	**2.6**
scb2	5.8	5.3	**5.0**
scb3	15.7	25.6	**9.1**
scb4	14.0	16.9	**13.2**
scb5	**10.6**	11.3	10.9
scb6	**11.3**	12.7	12.8
scb7	**16.4**	20.2	19.8
scb8	**51.7**	89.2	59.5
scb9	**54.0**	98.6	80.0
cpu-time *scb8*	120 s	366 s	39 851 s
cpu-time *scb9*	135 s	440 s	34 709 s
ratio	1	:3	:300

TABLE IV
COMPARISON OF RESULTS FOR CIRCUIT *PRIMARY1*

Comparison Criterion	Placement Method	
	Min-Cut	GORDIAN
Area of routed circuit/mm²	39.5	41.7 (1.06)
Wiring length in layer 1/mm	768	841 (1.10)
Wiring length in layer 2/mm	576	553 (0.96)
Cpu time/s	558	163 (0.29)

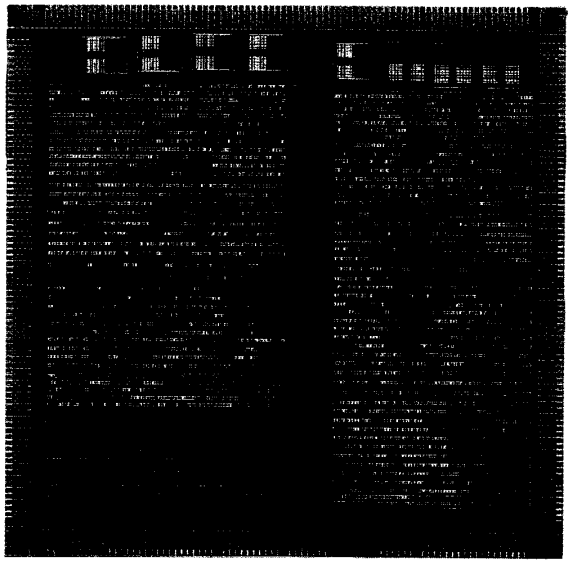


Fig. 10. Macrocell design with standard cell blocks *scb8* and *scb9*.

in layers 1 and 2, and CPU time needed by the placement methods measured on a Apollo DN4500 workstation running DOMAIN/IX. Both global and detailed routing was carried out by the same tools. Additionally in the GORDIAN column, the ratio of its result compared to min-cut is shown in parentheses. GORDIAN outperforms the min-cut based procedure more and more as the circuits become larger, needing much less CPU time.

TABLE V
COMPARISON OF RESULTS FOR CIRCUIT *STRUCT*

Comparison Criterion	Placement Method	
	Min-Cut	GORDIAN
Area of routed circuit/mm²	13.2	9.2 (0.70)
Wiring length in layer 1/mm	1018	558 (0.55)
Wiring length in layer 2/mm	632	362 (0.57)
Cpu time/s	2555	446 (0.18)

TABLE VI
COMPARISON OF RESULTS FOR CIRCUIT *PRIMARY2*

Comparison Criterion	Placement Method	
	Min-Cut	GORDIAN
Area of routed circuit/mm²	181	141 (0.78)
Wiring length in layer 1/mm	6823	4761 (0.70)
Wiring length in layer 2/mm	5226	3153 (0.60)
Cpu time/s	9457	912 (0.10)

TABLE VII
COMPARISON OF RESULTS FOR CIRCUIT *BIOMED*

Comparison Criterion	Placement Method	
	Min-Cut	GORDIAN
Area of routed circuit/mm²	104	68 (0.65)
Wiring length in layer 1/mm	9770	5232 (0.54)
Wiring length in layer 2/mm	5819	3123 (0.54)
Cpu time/s	40984	2494 (0.06)

TABLE VIII
COMPARISON OF SEA-OF-GATES RESULTS

Circuit				Weighted Estimated Wiring Length	
Name	#cells	#pads	#nets	GORDIAN	Min-Cut
sog1	682	44	820	**184 045**	212 258
sog2	1755	57	1809	**440 409**	687 769
sog3	2292	153	2775	**598 992**	744 466
sog4	2669	86	3128	**522 651**	609 557
sog5	4274	179	4580	**970 166**	1 260 784
sog6	6112	163	7261	**930 663**	1 803 165

7.2. *Sea-of-Gates Circuits*

GORDIAN has also been compared to a min-cut placement procedure for sea-of-gates circuits [33]. Table VIII confirms the results obtained with standard cell circuits. GORDIAN performed better in all cases. The results are given in terms of weighted estimated wiring length, measured as Manhattan-metric minimum spanning trees. The improved placement leads to lower wiring densities and lengths, which results in a drastically reduced number of unrouted connections and reduced CPU-times due to less rip-up and reroute.

The placement of a 6112 cell sea-of-gates circuit *sog6* was obtained within 10 min on a 15 MIPS computer. This circuit consists of 28K random logic with 63% basic cell utilization. Two metal layers were used for routing. Fig. 11 shows the final placement of this circuit obtained after exhaustive slicing opti-

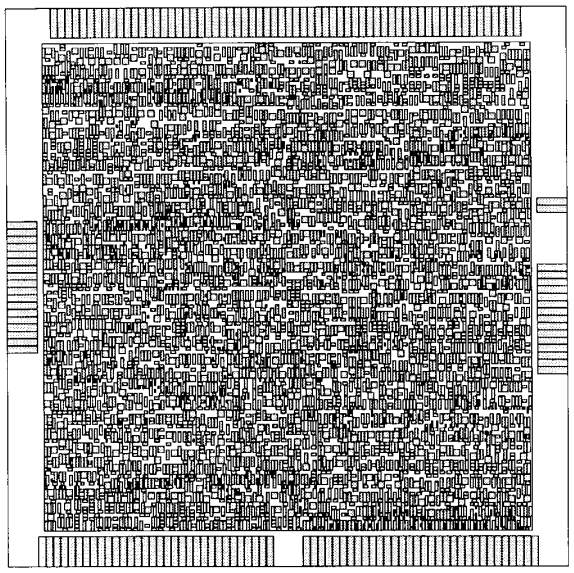


Fig. 11. Sea-of-gates circuit *sog6*: final placement.

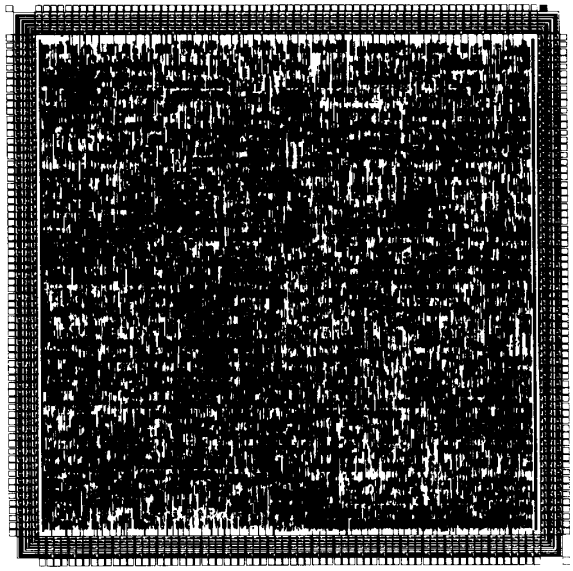


Fig. 12. Sea-of-gates circuit *sog6*: metal 2 routing.

mization with $k = 7$. In contrast to most of the available sea-of-gates design systems which place the cells in rows to create wiring channels, the approach taken here is to place the cells like for a huge macrocell circuit. The underlying slicing structure can hardly be detected in Fig. 11. This good area utilization is a result of the ESO procedure described in Section V-5.2. The final routing on the second metal layer is shown in Fig. 12. It shows a uniform distribution of the wiring density.

VIII. Conclusions

GORDIAN, a new placement method based on simultaneous quadratic programming combined with improved partitioning schemes and exhaustive slicing optimization, has been presented. Results obtained for large industrial designs substantiate distinct improvements in placement quality and computation time compared with state of the art placement tools. The global view of GORDIAN particularly pays off with increasing circuit size. Our experiments indicate that GORDIAN will be able to obtain high quality results with low computation times for circuits with tens of thousands of modules. To satisfy the high wiring requirements of such designs, our future work will concentrate on combining global placement with global routing. With an improved net modeling derived from the global routing, it will be possible to incorporate timing constraints during global placement.

Acknowledgment

The authors would like to thank M. Bartholomeus, U. Lauther, R. Dachauer, S. Mayrhofer, and W. Weisenseel of Siemens AG for their support in providing examples and comparing the experimental results. The helpful comments of the anonymous reviewers are also gratefully acknowledged.

References

[1] U. Lauther, "A min-cut placement algorithm for general cell assemblies based on a graph representation," in *ACM/IEEE Proc. 16th Design Automation Conf.*, 1979, pp. 1–10.

[2] A. E. Dunlop and B. W. Kernighan, "A procedure for placement of standard-cell VLSI circuits," *IEEE Trans. Computer-Aided Design*, vol. CAD-4, pp. 92–98, Jan. 1985.

[3] D. P. LaPotin and S. W. Director, "Mason: A global floorplanning approach for VLSI design," *IEEE Trans. Computer-Aided Design*, vol. CAD-5, pp. 477–489, Oct. 1986.

[4] P. R. Suaris and G. Kedem, "An algorithm for quadrisection and its application to standard cell placement," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 294–303, Mar. 1988.

[5] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell Syst. Tech. J.*, pp. 291–307, 1970.

[6] C. M. Fiduccia and R. M. Mattheyses, "A linear-time heuristic for improving network partitions," in *ACM/IEEE Proc. 19th Design Automation Conf.*, 1982, pp. 175–181.

[7] T.-K. Ng, J. Oldfield, and V. Pitchumani, "Improvements of a mincut partition algorithm," in *Proc. IEEE Int. Conf. CAD, ICCAD-87*, 1987, pp. 470–473.

[8] K. J. Antreich, F. M. Johannes, and F. H. Kirsch, "A new approach for solving the placement problem using force models," in *IEEE Int. Symp. Circuits and Systems, Proc. ISCAS*, 1982, pp. 481–486.

[9] G. J. Wipfler, M. Wiesel, and D. A. Mlynski, "A combined force and cut algorithm for hierarchical VLSI layout," in *ACM/IEEE Proc. 19th Design Automation Conf.*, 1982, pp. 671–676.

[10] C.-K. Cheng and E. S. Kuh, "Module placement based on resistive network optimization," *IEEE Trans. Computer-Aided Design*, vol. CAD-3, pp. 218–225, July 1984.

[11] K. M. Just, J. M. Kleinhans, and F. M. Johannes, "On the relative placement and the transportation problem for standard-cell layout," in *ACM/IEEE Proc. 23rd Design Automation Conf.*, 1986, pp. 308–313.

[12] R. Tsay, E. S. Kuh, and C.-P. Hsu, "PROUD: A fast sea-of-gates placement algorithm," in *ACM/IEEE Proc. 25th Design Automation Conf.*, 1988, pp. 318–323.

[13] R. H. J. M. Otten, "Eigensolutions in top-down layout design," in *IEEE Int. Symp. on Circuits and Systems, Proc. ISCAS*, 1982, pp. 1017–1020.

[14] J. P. Blanks, "Near-optimal placement using a quadratic objective function," in *ACM/IEEE Proc. 22nd Design Automation Conf.*, 1985, pp. 609–615.

[15] J. Frankle and R. M. Karp, "Circuit placements and cost bounds by eigenvector decomposition," *IEEE Int. Conf. on CAD, ICCAD-86*, pp. 414–417, 1986.

[16] J. M. Kleinhans, "Ein Plazierungsverfahren für den zellenbasierten Layoutentwurf hochintegrierter Schaltungen," Ph.D. dissertation, Inst. Computer-Aided Design, Dep. Elect. Eng., Tech. Univ. Munich, 1989.

[17] B. Preas and K. Roberts, presented at Physical Design Workshop, Hilton Head, SC, 1987.
[18] J. M. Kleinhans, G. Sigl, and F. M. Johannes, "GORDIAN: A new global optimization/rectangle dissection method for cell placement," in *Proc. IEEE Int. Conf. on CAD, ICCAD-88*, 1988, pp. 506–509.
[19] ——, "Sea-of-gates placement by simultaneous quadratic programming combined with improved partitioning," in *Proc. IFIP TC 10/WG 10.5 Int. Conf. on Very Large Scale Integration, VLSI '89*, 1989, pp. 445–454.
[20] A. A. Szepieniec and R. H. J. M. Otten, "The genealogical approach to the layout problem," in *ACM/IEEE Proc. 17th Design Automation Conf.*, 1980, pp. 164–170.
[21] R. H. J. M. Otten, "Graphs in floor-plan design," *Int. J. Circuit Theory Appl.*, vol. 16, pp. 391–410, 1988.
[22] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *J. Res. Nat. Bur. Stand.*, vol. 49, no. 6, pp. 409–436, 1952.
[23] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Baltimore, MD: Johns Hopkins University, 1985.
[24] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*. London, U.K.: Academic, 1981.
[25] L. P. P. P. van Ginneken and R. H. J. M. Otten, "Optimal slicing of plane point placements," in *ACM/IEEE Proc. 1st European Design Automation Conf.*, 1990, pp. 322–326.
[26] R. H. J. M. Otten, "Efficient floorplan optimization," in *Proc. Int. Conf. on Computers and Design, ICCD-83*, 1983, pp. 499–501.
[27] L. Stockmeyer, "Optimal orientations of cells in slicing floorplan designs," *Inform. Control*, vol. 57, pp. 91–101, 1983.
[28] G. Zimmermann, "A new area and shape function estimation technique for VLSI layouts," in *ACM/IEEE Proc. 25th Design Automation Conf.*, 1988, pp. 60–65.
[29] International Workshop on Layout Synthesis, Microelectronics Center of North Carolina, Research Triangle Park, NC, May 1990.
[30] M. Bartholomeus and R. Weismantel, "Simulated annealing for standard-cell placement: Introduction, implementation and results," in *Proc. of Operations Research, Ulm, West Germany*, 1989, pp. 171–190.
[31] E. Hörbst, C. Müller-Schloer, and H. Schwärtzel, *Design of VLSI Circuits Based on VENUS*. Berlin, Germany: Springer-Verlag, 1987.
[32] L. Schrader, N. Haff, and A. Weller, "An ADVANCELL 1.0 mainframe chipset—2.2 million transistors on 11 ICs," in *Proc. IEEE Custom Integrated Circuits Conf., CICC-90*, 1990.
[33] U. Lauther, "On automatic placement methods," Technical Rep., Siemens Corp. Res. Develop., Appl. Comput. Sci. and Software, Syst. Design Automation, (in German). Munich, Germany, Nov. 1988.

Jürgen M. Kleinhans received the Dipl.-Ing. degree and the Dr.-Ing. degree in electrical engineering from the Technical University of Munich, Germany, in 1982 and 1989 respectively.

From 1982 to 1989, he was Research Assistant at the Institute of Computer-Aided Design, Department of Electrical Engineering, Technical University of Munich. Since 1989, he has been with the Systems Design Automation Group, Siemens Corporate Research and Development, Munich, Germany. His research interests include computer-aided design of integrated circuits and systems, with emphasis on layout synthesis and on methods related to floorplanning, supporting a system developer early in the design process.

Georg Sigl received the Dipl.-Ing. degree in electrical engineering from the Technical University of Munich, Germany, in 1987. Since 1987, he has been working towards the Dr.-Ing. degree at the Institute of Computer-Aided Design, Department of Electrical Engineering, Technical University of Munich.

Currently, he is a Research Assistant with the Technical University of Munich. His research interests include the computer-aided design of integrated circuits and systems, with particular emphasis on layout synthesis.

Frank M. Johannes received the Dipl.-Ing. degree in electrical engineering from the Technical University of Karlsruhe, Karlsruhe, Germany, in 1968, and the Dr.-Ing. degree from the University of Erlangen-Nürnberg, Erlangen, Germany, in 1973.

From 1968 to 1975, he was with the Regional Computing Center in Erlangen, working in the fields of computer networking and computer-aided software engineering. In 1976, he joined the Institute of Computer-Aided Design, Department of Electrical Engineering, Technical University of Munich, where he is currently heading a layout research group. His research interests are in the area of computer-aided design of electronic circuits and systems, with emphasis on layout synthesis.

Kurt J. Antreich (M'69–SM'78) received the Dipl.-Ing. degree from the Technical University of Munich, Munich, Germany, in 1959, and the Dr.-Ing. degree from the Technical University Fridericiana Karlsruhe, Karlsruhe, Germany, in 1966.

He joined AEG-Telefunken in 1959, where he was involved in computer-aided circuit design for telecommunication systems. From 1968 to 1975 he was head of the Advanced Development Department of AEG-Telefunken, Backnang, Germany. Since 1975, he has been a Professor of Electrical Engineering at the Technical University of Munich, Munich, Germany. His research interests are in computer-aided design of electronic circuits and systems, with particular emphasis on circuit optimization, testing, layout synthesis, and synthesis of digital systems.

Dr. Antreich was the Chairman of the NTG Circuit and Systems Group from 1972 to 1974, and a member of the executive board of the NTG from 1979 to 1982, and chairman of the NTG CAD Group from 1985 to 1989. He received the NTG prize paper award in 1976. He is a member of the ITG (old name: NTG) and the GI (Gesellschaft für Informatik, Germany).