A TREE-BASED ALGORITHM FOR COMPONENT PLACEMENT

A thesis submitted to the

University of Manchester

for the degree of Doctor of Philosophy

in the Faculty of Science

by

Maria Rosália Dinis Rodrigues

Department of Computer Science

University of Manchester

May 1986

ABSTRACT

The subject of this thesis is the development of a general method for the automatic placement of components on a circuit board. The layout of integrated circuits motivated the use of hierarchical techniques as a means of dealing with increasing circuit complexity. A new approach to hierarchical placement is suggested in this thesis, based on a tree structure which embodies an adopted set of circuit properties and objective functions.

The placement problem is here defined in mathematical terms and a formulation is proposed for its most widely accepted figures of merit. Three placement objectives are selected and a graph theoretical study is presented, which investigates the correspondence between those objectives and the structure of a binary tree.

A new placement method is proposed in this thesis, designed in accordance with the suggested fully hierarchical philosophy. The method consists of two main steps: the building of the tree structure representing the circuit hierarchy and its subsequent embedding on different board environments. A set of algorithms is presented, for the tree-building and tree-mapping on two basic types of boards: regularly structured and continuous plane.

A practical implementation showed that the method is fast and can generate placement solutions which are comparable with those obtained manually, both in terms of measurable circuit objectives and observed routability.

D E C L A R A T I O N

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

ACKNOWLEDGEMENTS

I acknowledge with deep gratitude the advice, inspiration and interest provided by Dr. Hilary J. Kahn and Dr. Chris C. Kirkham, my supervisors.

I would also like to record my thanks to the colleagues and staff at the Department of Computer Science for all the helpful discussions and suggestions. To my friends at the Departamento de Matemática in Coimbra (Portugal), and particularly to Prof. Fernanda A.A. Neves de Oliveira, I express my gratitude for continuous encouragement and support.

I am also grateful to the portuguese Instituto Nacional de Investigação Científica for the financial support and to Maria do Céu Chaves who typed this thesis with so much care and patience.

To my parents,

with deep affection

and gratitude.

CONTENTS

Page

CHAPTER I - THE PLACEMENT PROBLEM

1. INTRODUCTION 1
2. LAYOUT AUTOMATION 2
3. METHODOLOGY OF LAYOUT AUTOMATION 3
4. HIERARCHICAL APPROACH 5
5. PHYSICAL HIERARCHY 7
6. LAYOUT ALGORITHMS 9
7. THE PLACEMENT PROBLEM 12
8. CIRCUIT REPRESENTATION 15
9. OBJECTIVE FUNCTIONS 20
   9.1. Minimal distance objectives 20
   9.2. Even distribution objectives 24
10. CONCLUDING REMARKS 26

CHAPTER II - PLACEMENT METHODS

1. INTRODUCTION 28
2. CLASSICAL METHODS 29
   2.1. Initial placement 29
   2.2. Minimal distance methods 30
   2.3. Analytical methods 32
   2.4. Even distribution methods 33

Page

3. HIERARCHICAL APPROACH 34

3.1. Top-down placement 34

3.2. Bottom-up placement 36

4. THE USE OF GRAPHS 38

4.1. Topological approach 38

4.2. Polar graph concept 41

4.3. Symbolic layout 42

4.4. Compaction 43

5. A NEW PHILOSOPHY 45

6. PROPOSED METHOD 47

7. SCOPE AND RESTRICTIONS 51

8. ABSTRACT OF THE FOLLOWING CHAPTERS 53

CHAPTER III - GRAPHS, PARTITIONING AND TREES

1. INTRODUCTION 55

2. NOTATION AND CONCEPTS 56

3. THE PARTITIONING PROBLEM 60

4. A LOOK AT PREVIOUS WORK ON PARTITIONING GRAPHS 63

5. ESTABLISHING OBJECTIVES 65

5.1. Minimal total distance 65

5.2. Even distribution of wires 67

5.3. Balancing the tree 69

5.4. Concluding remarks 70

6. TREE BUILDING ALGORITHM 71

6.1. The selection of S 72

6.2. The choice of C 72

6.3. Algorithm 73

6.4. A practical example 74

6.5. Complexity 79

CHAPTER IV - MAPPING THE CONNECTIVITY TREE

1. INTRODUCTION 81

2. BASIC PROCESS 82

2.1. Areas 82

2.2. Permutation 84

2.3. Squared distance 86

2.4. The method 90

3. REGULARLY STRUCTURED BOARDS 92

3.1. Space conflict 93

3.2. Proportionality 95

3.3. Preplacement 96

3.4. Algorithm 97

4. GENERAL CASE 99

4.1. Evaluation of areas 99

4.2. Preplacement 100

4.3. Space fitting 101

4.4. Algorithm 105

Page

CHAPTER V - IMPLEMENTATION AND RESULTS

1. INTRODUCTION 107
2. IMPLEMENTATION 108
3. REGULARLY STRUCTURED BOARDS 109
   3.1. Tree adjustments 110
   3.2. Preplacement 116
4. GENERAL CASE 121
   4.1. Tree adjustments 124
   4.2. Tracking 129

CHAPTER VI - CONCLUSIONS

1. A REVIEW OF CONCEPTS 134
2. SUGGESTIONS FOR FURTHER RESEARCH 137

REFERENCES 141

CHAPTER I

THE PLACEMENT PROBLEM

1. INTRODUCTION

In this introductory Chapter we place the Placement Problem within the general field of Layout Automation. Fundamental layout concepts are introduced, the hierarchical design philosophy is discussed and the different layout stages are briefly specified.

The Placement Problem is then defined in mathematical terms and related formulations are discussed. This section is followed by a survey of the use of graph theoretic structures in circuit representation.

The final paragraphs comprise a study of the definition of objective functions for the Placement Problem.

2. LAYOUT AUTOMATION

The *layout problem* can be formulated in the following way: given a number of elements (components) and a list of connections to be made between these elements, position the elements on a plane and generate the physical connections on one or more planes, taking any specific constraints into account.

This involves a large range of subjects such as: the generation of electronic circuit drawings, logic diagrams, flowcharts, the layout of printed circuit (PC) boards, integrated circuit (IC) masks, etc..

Automatic layout has been used in earlier small and medium scale circuits as a means of speeding up the layout process as well as relieving the human designer of repetitive and error prone tasks. With the ever increasing complexity of very large scale integration (VLSI), layout automation has become a real necessity. The increasing demand for custom large scale integration (LSI) chips which are to be produced in small quantities, has made this necessity even more important as a means of design cost reduction. This cost is based on two factors: the design time and the number of errors per design cycle. Therefore layout automation is aimed at the reduction of both.

The term *layout automation* has been used to specify a wide range of different degrees of assistance provided by the computer. At a lower degree, manual layout is often supported by machine digitizing and editing in the final phase of the design sequence. This approach can produce very dense layouts, but at the expense of an enormous design time and high risk of errors. In an intermediate or semiautomatic philosophy, automatic layout routines (e.g. placement and routing) are used to aid and even to replace the designer. He supervises the call of procedures, and can either accept the result or try a different input. At the extreme level of the automatic design spectrum, full use is made of the analyzing and decision making capabilities of the computer. Dedicated software is applied to automatically process the design through each step from raw logic data to final routing. The output is quite predictable in performance and production costs are minimal both in design time and error checking.

3. METHODOLOGY OF LAYOUT AUTOMATION

Printed circuit board layout has received extensive attention from industry, leading to the development of several algorithms and methods that allow automated PC board layout. Most of these algorithms are particularly suited for *regularly structured boards* with fixed locations (*slots*) for uniformly shaped components. This approach was introduced to simplify the problem algorithmically and also because regularly structured PC boards are easier to manufacture. One disadvantage is that this restriction can lead to bad utilization of the available space.

The conventional approach has been proved inadequate for certain types of layout problems, especially the case of LSI circuits. These are frequently composed of elements of very different size and shape and it is important to minimize the total chip area.

With the necessity for LSI design automation, a number of layout concepts have been introduced and extensively developed, such as standard cell, polycell, general cell, gate array, etc. *Standard cell* assemblies have a regular organization and so are particularly suited for the layout algorithms, most of the effort being spent in the design and maintenance of the standard cell library. In the *polycell* approach, cells have all approximately the same height and are placed in regular rows. *General cells* are not restricted in size or shape and may be assigned to any position on a *continuous plane*. A *gate array* is a predefined pattern of basic devices and gates which may be interconnected during the final manufacturing states in order to achieve the specified chip function.

The algorithms developed under a *regular structure* methodology for the placement and the routing are therefore only applicable to standard cell and gate array environments. They cannot be used as such for general cell assemblies. In this thesis, the term *general placement* will refer to the type of environment where components of different size and shape are to be placed on a continuous plane without predefined locations.

Under both approaches, however, it may happen that the position of particular modules is totally or partially specified. The components subject to this type of design requirement will be referred to as *preplaced*. An example of partial preplacement is the positioning of *edge connectors* along one (or more) of the board boundaries.

The layout of general cell assemblies has brought up the need for a different approach to layout automation. Furthermore with the constant increase in circuit complexity, new methods had to be devised to handle the enormous amount of data. The natural solution was the subdivision of the layout problem into several hierarchical levels [MC80]. A general cell assembly may thus consist of combinations of standard cell assemblies, standard cells or lower level general cell assemblies. The use of a recursive process reduces the circuit complexity to a level consistent with the available design capacity.

4. HIERARCHICAL APPROACH

A hierarchical approach itself is not a new idea. Software designers have learned how to deal with increasingly complex programs by developing a structured programming methodology. The human hardware designer also, often performs his task in a hierarchical fashion by repeatedly splitting the problem into smaller manageable units.

Starting with a set of given specifications, the designer makes a number of design decisions, both in terms of logical design and physical design. Those decisions can take one of two major forms: *top-down* decomposition of modules into less complex ones and *bottom-up* combination of building blocks into larger ones. At some stage in the process, the modules will have to be mapped into some of the building blocks in such a way that this mapping may result in a correctly operating design.

The design execution is therefore a combination of top-down and bottom-up processes that are happening concurrently in the designer's mind until he reaches a final acceptable design.

The design process is characterized by three cooperating tasks:

. *Behaviour design* - where the initial specification is decomposed into subproblems, and possibly refined.

. *Structural design* - where blocks or modules are realized by the interconnection of more primitive modules.

. *Physical design* - where the design is implemented in a given technology.

In most cases behaviour and structural designs are concurrently developed, whereas the physical design process is done separately. Technological constraints may however have a very strong influence on the behaviour or the structural designs of a system. As an example, a situation may arise where an application requires customized random logic which is not found in the available standard cell library. A new cell must then be designed or the whole design process iterated in order to meet the available design resources. A recent work in this field [HG85] describes a new type of design phylosophy where the three basic design tasks are performed concurrently. The given functional specification of a module is, through a sequence of transformations, decomposed into geometric cells which fit together in a given silicon area.

Three major hierarchies may thus be associated with the digital system design procedure: a behaviour hierarchy, a structural hierarchy and a physical hierarchy. The mapping of a behaviour into a structural hierarchy is usually known as the *logic design process*. In actual practice this is an iterative task done mainly in the designer's mind without the help of design automation tools. Few of the existing logic design systems are supported by automatic or computer-aided techniques to reduce the time required to perform an error--free mapping of behaviour into structure.

The mapping of a structure into physical hierarchy is usually known as the *physical process*. Currently, and for integrated circuit design in particular, this process is supported by a collection of software packages for logic simulation, automated placement and routing, circuit analysis and design rule verification. However, very few of the existing physical design systems are consistent with a wholly hierarchical approach.

5. PHYSICAL HIERARCHY

In order to package a system the designer has at his disposal a hierarchy of cabinets, racks and printed circuit boards. This physical hierarchy is extended to each integrated circuit on a PC board. In IC design he may use a succession of supercells, macrocells, simple cells and transistors. In the course of designing large scale integrated circuits, the human designer naturally adopts an approach of breaking down the design into blocks, which are successively subdivided until the transistor level.

Several automated hierarchical design systems have been developed, especially for VLSI circuits. Their major advantage is the reduction in design time, while keeping sufficiently detailed information to allow concise prediction of the behaviour of the system.

A major problem in hierarchical design is the mapping from a structural into a physical hierarchy. One possible solution could be a direct correspondence, i.e., to make every structural module the same as every physical module. This solution may however lead to inefficient area utilization. A successful mapping should be able to look-ahead over one or more levels of the physical hierarchy.

The physical process is essentially a partitioning problem in the graph theoretical sense. Given detailed information about the connectivity of a system, as provided by the structural design, subdivide it into subsystems of a given maximum size. The successive partitioning of the subsystems generates a tree structure where the whole system is represented by the tree root and the other nodes correspond to the subsystems, the leaves being the simplest components considered.

In this tree structure depicting the physical hierarchy of a system, it may happen that a particular node represents a subdivision into elements which belong to the same plane surface (e.g. the passage from PC boad to IC, IC to general cell or general cell to standard cell). According to our previous definition in paragraph 2, that partition corresponds to a layout problem.

The layout may, in turn, be expressed in a hierarchical fashion. That corresponds to the expansion of a single level partition into several new partition levels. The number of such levels is a direct function of both the number of elements involved and the degree (number of subsets) of each partition.

In general the given layout plane is rectangular in shape, and so are the components to be positioned on it. The partitioning process is then the same at each level, representing the assignment of groups of elements to rectangular board areas.

In order to evade the problem of dissecting a rectangle into three or more predefined areas, binary partitions are frequently associated with the layout problem. However, most of the existing layout methods which are based on partition are not of a hierarchical nature. Those partition processes are usually based on iterative swapping of pairs of elements across boundary lines on the board. Only recently, attempts have been made to elaborate the layout hierarchy first, and then to embed it on the board surface [Ri84] [0185]. In an integrated layout methodology, the partitioning technique (tree building) will be essentially the same for all types of layout. The main difference will be in the embedding of the tree according to the particular type of environment: regular structured board or continuous plane.

6. LAYOUT ALGORITHMS

One major objective in the layout design of a circuit on a PC board is to complete all the necessary connections in the routing phase. The connection success depends on the router being used but is also largely determined by the effectiveness of the preceding steps.

Before a router is used, the following steps are expected to be done automatically in such a way as to produce high routability: board partitioning, component assignment, placement and gate assignment.

The allocation of a circuit to several boards so as to minimize the interboard connections is a typical partitioning problem. Similarly, component assignment is a partition of the circuit elements (logical gates) into subsets to be assigned to the components, in such a way that the intermodule connections are minimized.

Placement defines the positions of the modules on the board. How to state the Placement Problem is what we are going to analyse in detail in the remaining paragraphs of this Chapter.

The step of gate assignment is essentially a placement at the component level. After each component is positioned, this process consists of the allocation of a predefined group of gates to a set of logically equivalent gate sites.

In the routing phase of a PC board and before the actual wire layout, four basic steps are usually followed which may to a large extent contribute to the intended routing success. Those steps are: the determination of precisely which wires are to be laid out, the assignment of connections to specific pins of a component, the allocation of each wire to one of the available board layers and the establishment of the order in which the wires on each layer should be processed.

Some routing methods of a structured nature have been proposed, particularly concerning the ordering problem. Usually not related to the previous layout steps, they define their own hierarchy of areas on the board which are successively routed and then combined in a bottom-up fashion.

In integrated circuit layout the main objectives are 100% complete routing and also the minimization of the total chip area. The type of effective computer design aids that can be adopted in the pursuit of those objectives is largely influenced by the methodology in use: standard cell, polycell or general cell approaches.

In the standard cell and polycell approaches the fundamental phases are still placement and routing, performed usually in separate operations. Optimization of the placement can be done very easily by interchanging cells in different rows or in the same row. Routing can be performed effectively with a fast channel router.

In the layout of general cell assemblies however, the use of hierachical methods becomes a real necessity. The circuits are extremely complex and comprise elements of dissimilar size like memory structures, PLA's or entire standard cell assemblies already laid out.

By using a hierarchical design method, the circuit complexity can be spread out in as many levels as necessary to make it consistent with the available design capability. Only one level is processed at each time and the basic algorithms are the same for each recursion level. Placement and routing are optimized at each level before proceeding to the next one.

Another important characteristic of general cells is that input--output pins are not restricted to any particular cell side so that routing channels are not predefined. With a structured approach, it becomes possible to optimize the pin positioning of a cell by the interconnections with the others at the same level, and at a lower level adjust the layout of that cell so as to be optimal for that pin positioning.

In most of the existing general cell layout systems, routing is performed in three stages. The first one is the definition of the routing channels, usually done in parallel with placement which enables a correct estimate to be made of channel widths and consequent area optimization. In the second stage, or global routing, nets are loosely assigned to the routing channels. It is only in the third stage that the nets are finally assigned to a specific track in the routing channel.

Routing is, in any type of layout, a costly process requiring an enormous amount of computer time. Its cost is primarily dependent on circuit complexity and on the wirability provided by the placement technique used. An optimal placement is critical to successful cost-effective routing.

The routing problem has been intensively studied and methods are known which guarantee a 100% completion under adequate conditions.

Rather than improving the already complex routing scheme, we believe that the major factor in improving the wirability of a layout system, is the enhancement of the placement phase. This was the main motivation for the research reported in this thesis.

7. THE PLACEMENT PROBLEM

Let $C = \{C_i | i=1,...,n\}$ be a given set of components. Every component has a number of terminals or pins to which interconnections are made.

A *net* is a collection of terminals (of the same or of different components) that are connected to each other. Let $N = \{N_k | k=1,...,m\}$ be a given set of nets.

A *physical circuit* can then be represented by a system denoted (C,N) of components and nets.

Given a rectangular area or *board* B and a circuit (C,N), the *Placement Problem* consists of assigning to each $C_i \in C$ a unique position on B, defined by its Euclidean coordinates (x_i,y_i), in such a way that some objective is optimized.

The placement problem cannot be "solved", in the mathematical sense, due to its combinatorial complexity and also to the difficulty in defining objective functions. The question is, in practice, approached through other problems of related formulation and easier interpretation. For that reason, a number of more or less acceptable simplifications are usually introduced.

The most commonly used simplifications of the placement problem are the following four:

. Assume that the only objective is to minimize the total wirelength.

. Reduce each component to a point (its geometrical centre) where all pins are identified.

. Represent all nets in terms of one-to-one connections.

. Assume that the board has a finite number of locations, to which components can be assigned. All components are considered equal in size and can take any location.

Under those assumptions, the total wirelength can be written as:

$$\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} c(i,j).\ \mathrm{dist}\ [s(C_i),\ s(C_j)]$$

where $c(i,j)$ is the number of wires between components C_i and C_j, and $\mathrm{dist}[s(C_i),\ s(C_j)]$ is the distance between the slots, or locations, to which C_i and C_j have been assigned. This function has to be minimized over all possible permutations of each component to each slot.

This problem is known as the Quadratic Assignment Problem, very often mentioned in association with the Placement Problem. It should however be noted that the quadratic assignment is, in itself, a very complex combinatorial problem. It has been shown by Garey et al. [GJ74] that the quadratic assignment belongs to a class of problems called *NP-complete*, either all or none of which are solvable in polynomial time. Since many infamous combinatorial problems have been proved to be NP-complete, the latter alternative seems far more likely.For that reason, enumerative optimization methods and approximation algorithms are frequently used.

From the above discussion and the rigidity of the former assumptions, one must question the feasibility of reducing the placement problem to the quadratic assignment problem. Even if an optimum aolution to the associated quadratic assignment problem was found in a reasonable amount of computation time, that one would not necessarily be an optimum solution to the original placement problem.

To assume that minimal total wirelength is the only objective function may not necessarily increase routability, because it will tend to create areas of wiring congestion.

The fourth assumption is also particularly restrictive. Only in the case of regularly structured boards with fixed component locations can permutations be considered. In most of the cases, the board is a continuous plane without predefined slots. It should also be noted that this restriction evades the complex problems of rectangular cutting and packing, often encountered in the placement of general cells.

The second and third assumptions are made in order to represent the circuit in terms of a graph.

8. CIRCUIT REPRESENTATION

The definitions of the last paragraph reveal two different aspects of the circuit layout problem. The "topological" aspect relates primarily to the manner in which components are interconnected. Further topological information may include the order in which the terminals of a component appear on its outer boundary, interchangeability of terminals and the requirement that the external connections have to appear on the outside boundary of the circuit in a prespecified order.

The "geometrical" aspect of the circuit layout problem is primarily related to parameters that can be measured. Distances, size of individual components, thickness of conductor lines and the size of a printed circuit board or an integrated circuit chip are examples of geometrical parameters.

A physical circuit (C,N) as defined, comprises topological information about a set of components and the way they are related by a set of nets. On the circuit board B, components are assigned to Euclidean coordinates and related by distances on the plane. Hence the effect of the placement process is, in some sense, to embed the circuit's topology in the board's geometry.

Placement algorithms require, as a framework, a convenient circuit representation or *model*. An ideal circuit model should represent, as faithfully as possible, all the topological aspects and also take the geometrical parameters into account. In practice, this objective is limited by the actual functionality of the model. The efficiency of a placement algorithm may however be determined by the correct choice of a circuit representation.

The definition of physical circuit is closely related to the mathematical concept of a hypergraph [Be70]. Given a finite set $X = \{x_1, x_2, \ldots, x_k\}$ and a family $E = \{E_i | i \in I\}$ of subsets of X, the pair (X,E) is called a *hypergraph* if $E_i \neq \phi$ $(i \in I)$ and $\bigcup_{i \in I} E_i = X$.

The elements of X are the *nodes* and the sets $E_i \in E$ are the *edges* of the hypergraph (X,E).

A physical circuit (C,N) has the structure of a hypergraph, where X is the set of all component terminals which are connected and E is the set N of all nets. In other words, terminals are the nodes and nets are the edges of the hypergraph representing the circuit. Since components are sets of terminals, they are also hypergraph edges. Fig. 1 shows a circuit and its hypergraph representation.

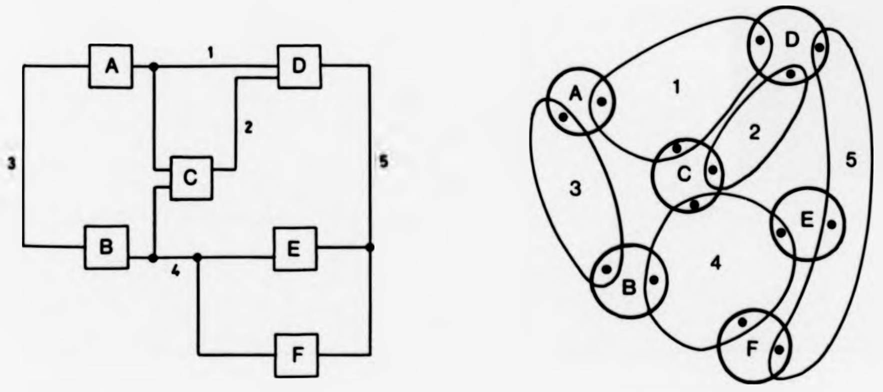


FIG. 1 - A circuit and its hypergraph representation

Hypergraph models have been used in association with the circuit partiton problem [La73], but are seldom related to the circuit layout problem. VanCleemput [Va76] developed hypergraph models for the circuit layout problem, that are able to represent ordering of terminals and other properties of physical circuits.

When all edges in a hypergraph are of cardinality 2, it becomes a simple *graph*. Graph models have been widely used in several areas of the design automation of digital systems, particularly in the circuit layout problem. They are easy to program and can be made to represent different types of circuit reflecting their topological and physical properties. A very concise mathematical formulation on the use of graph models for the circuit layout problem was established by VanCleemput [Va76]. Circuit properties were analysed in detail and different representations were developed and discussed.

The graph in Fig. 2.a) represents the classical "component-to-node, connection-to-edge" model for the sample circuit in Fig. 1. Components are represented by single nodes and each net is a *complete* graph, i.e. there is an edge joining every pair of nodes belonging to the net. The number of edges between two components therefore equals the number of nets which are common to them.

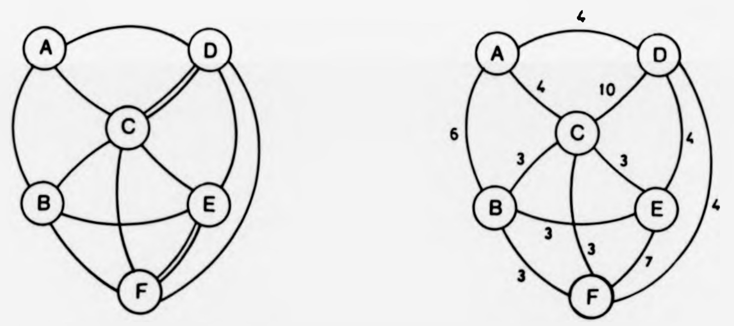


FIG. **2** - Graph model of the circuit in Fig. 1, with a) w=1 and b) w=2/n.

Since, in this representation, all edges were given the same weight (w=1) and each net is connected by a total of $\frac{n(n-1)}{2}$ edges, there is an overestimation of the high-order nets when compared to the low-order ones. A way to balance this disproportion [HW76] is to give the weight w = 2/n to each edge in a net of order _n_. In that case, the total weight becomes

$$\frac{n(n-1)}{2} \cdot \frac{2}{n} = n-1$$

which is the length of the minimum spanning tree connecting _n_ nodes. Fig. 2.b) shows this last representation with all weights multiplied by a factor of 6, for simplicity.

Representing nets in terms of complete graphs seems an adequate model for placement algorithms because the decomposition into single connections can be done in the course of the layout. The problem of net decomposition can be formulated as finding a spanning tree in a complete graph of _n_ nodes and, as it was first proved by Cayley in 1897, there are n^{n-2} possible decompositions [Be7C]. It would be premature to establish a particular one, prior to the actual placement of the components. The decomposition may depend on the specific type of net (chain, star, cycle,...) and also on the techonology in use. As an example, the circuit design in ECL and other fast logics may require a resistor next to the terminal module of each net. In this case, it is desirable that the decomposition of nets (and identification of terminal modules) be performed subsequently to the placement of components.

The basic representation of components by single nodes, can easily be adapted to include the cyclical order of terminals and other topological properties. Geometrical information, like the physical dimensions of each component, can be attached to the data structure representing the node and taken into account by the placement algorithms.

A simple model, like the one described, also provides a convenient representation for other types of design requirements. In addition to the basic distribution of weights, determined by the order of the nets, artificial weights can be imposed on specific edges representing vital connections. An extra measure of connectivity can thus be introduced between components belonging to a functional block or forming a loop which must be kept together for maintenance and timing reasons.

Generally speaking, circuit models based on graph theoretic structures supply a convenient framework for the placement algorithms. A variety of models have been developed [Va76], which are able to represent a wide range of circuit properties. According to the specific problem, a model can be organized as the one that best reflects the behaviour of the techonology under consideration and/or the expected properties of the final layout.

9. OBJECTIVE FUNCTIONS

The goodness of a placement is ultimately determined by how efficiently it can be routed. Routability is however an intangible objective, depending also on other factors besides placement, such as the particular type of circuit and the routing system which is to be applied afterwards. This fact explains the diversity of objectives and properties that is found in the literature reporting the different placement techniques. In particular, when a placement scheme is to be incorporated into an existing routing system, it naturally tends to accomplish the objectives that most adequately model the wiring features of the routing system.

There is nevertheless a general consensus on a number of properties, that practice has shown to be strongly related to circuit routability. Those are measurable properties that can be evaluated independently of the technology in use and the behaviour of any routing scheme.

The most widely accepted figures of merit for a good placement can be grouped into two classes which we will name *minimal distance* objectives and *even distribution* objectives. To the first class belong all classical objective functions such as the minimal total distance. Among the even distribution objectives is the minimization of the number of crossovers through boundary lines.

9.1 - Minimal distance objectives

Let (C,N) be a given physical circuit, whose set of components $C = \{C_i \mid i=1,\ldots,n\}$ is to be assigned to uniquely defined positions on a board B. The position of component $C_i \in C$ will be defined by the plane coordinates (x_i,y_i) of its centre point.

In addition, let d be some metric relating any two points in the plane such that

$$d_{ij} = d[(x_i,y_i) , (x_j,y_j)].$$

The set $\{d_{ij}\}_{i,j=1,...n}$ has the structure of a symmetric matrix with zeros along the main diagonal, and will be referred to as the *distance matrix*.

The most commonly used metrics are: the Euclidean distance

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

and the rectilinear distance

$$d_{ij} = |x_i - x_j| + |y_i - y_j|.$$

It is also assumed that every circuit net $N_k \in N$ has already been represented in terms of single edges. The number of edges between components C_i and C_j, possibly adjusted by some weighting factor as seen in the last paragraph, is said to be the *connectivity* c_{ij} between those components. The *connectivity matrix* $\{c_{ij}\}_{i,j=1,...,n}$ is also a symmetric one and, strictly for the present purpose, all the elements c_{ii} can be made equal to zero.

Let S be the set of all possible solutions of this placement problem. Given a nonnegative integer k and a solution $s \in S$, we define the *k-th order moment* of s as follows

$$M_k(s) = \sum_{i,j} c_{ij} \, d_{ij}^k(s) \quad .$$

where the sum is taken over all i,j such that $1 \leq i < j \leq n$.

In the particular case of $k = 0$, the expression becomes a constant,

$$M_0 = \sum_{i,j} c_{ij}$$

equal to the total connectivity of the circuit.

When $k=1$, the first moment is given by

$$M_1(s) = \sum_{i,j} c_{ij} \cdot d_{ij}(s)$$

that is, the total distance of placement s. The minimization of $M_1(s)$ over the set of all solutions $s \in S$ is known as the quadratic assignment problem.

If $k=2$, the second moment is determined as

$$M_2(s) = \sum_{i,j} c_{ij} \, d_{ij}^2(s).$$

The choice of $M_2(s)$ rather than $M_1(s)$ as an objective function for the placement problem, may bring considerable advantages. From a computational point of view, when the used metric is the Euclidean distance, it saves the square root evaluation repeatedly required in a placement algorithm. On the other hand, a solution obtained on $M_2(s)$ tends to produce fewer long distances, when compared with the minimal total distance solutions. This fact is to be expected when one evaluates the variance of the distribution of distances from the mean value. The value of the average distance in solution s, denoted $\bar{d}(s)$ is given by

$$\bar{d}(s) = \frac{\sum_{i,j} c_{ij}\; d_{ij}(s)}{\sum_{i,j} c_{ij}} = \frac{M_1(s)}{M_0}$$

The mean square deviation from the average of the distances is the number $\sigma_d^2(s)$ defined by

$$\sigma_d^2(s) = \frac{\sum_{i,j} c_{ij}\; (d_{ij}(s) - \bar{d}(s))^2}{\sum_{i,j} c_{ij}}$$

After some simple algebraic manipulation, this can be written as

$$\sigma_d^2(s) = \frac{M_0 \cdot M_2(s) - M_1^2(s)}{M_0^2} = \frac{M_2(s)}{M_0} - \bar{d}^2(s).$$

Since M_0 is a constant, there exists a close relation between the minimization of $M_2(s)$ and of $\sigma_d^2(s)$, i.e. a better uniformity of the produced distances. It should however be noted that the optimal solutions obtained on $M_2(s)$ and on $M_1(s)$ do not necessarily coincide. The most convenient approach might be to choose, among the best solutions in $M_1(s)$, the one with smaller $M_2(s)$ and consequently lower $\sigma_d^2(s)$.

For higher values of k, the k-th moment $M_k(s)$ is related to yet another frequently used objective of the placement problem, namely the minimization of the longest distance. In a given solution $s \in S$, the longest distance L(s) between connected components, is defined as

$$L(s) = \max_{i,j} \{\delta_{ij}\; d_{ij}\} \qquad \text{where} \quad \delta_{ij} = \begin{cases} 0 \text{ if } & c_{ij} = 0 \\ 1 \text{ otherwise}. \end{cases}$$

It has been proved by Steinberg [St61] that L(s) can be regarded as a limiting case of $M_k(s)$, i.e. there exists a K such that for all $k \geq K$, the optimal solution on $M_k(s)$ is also the optimal solution on L(s).

9.2 - Even distribution objectives

Another class of objectives has been introduced, suggested by two practical observations [Br77], namely:

. successful routing is dependent on the density of interconnections.

. some areas of the board are usually more dense than others.

It has also been observed that minimal total distance may not be a good criterion to use in order to increase routability, since heavily connected components tend to be clustered together, therefore creating areas of wiring congestion.

In order to enable a measuring of the density of interconnections on the board, a new concept was introduced and named *boundary line* or *cut line*. The board is artificially divided by equidistant horizontal and vertical lines, generating cells that approach the size of individual components. Every single connection between a pair of components will cross a number of boundary lines. For the set of all connections in the circuit and assuming that the shortest routes are taken, the amount of crossovers at each line can thus be determined.

Let x_ℓ be the number of crossovers at an individual boundary line ℓ. Over the set of all possible solutions $s \in S$, the first objective function one might consider is

$$X_1(s) = \sum_{\ell} x_{\ell}(s)$$

i.e., the total number of crossovers for all boundary lines $\ell = 1,\ldots,m$.

By taking the cell width as measuring unit, it can be easily proven that $X_1(s)$ equals $M_1(s)$ when the rectilinear distance is considered. Using $X_1(s)$ as an objective function will not, for that reason, lead to better results than the classical minimal distance objective.

As suggested by Wang [Wa80]: it is necessary not only to minimize the number of crossovers, but also to ensure a uniform distribution of these crossovers over the set of all cut lines.

For that effect Wang defined a new objective function that, in our own notation, is written as

$$X_2(s) = \sum_{\ell} x_{\ell}^2(s).$$

Again, by evaluating the variance $\sigma_x^2(s)$ of the number of crossovers from the mean value $\bar{x}(s) = X_1(s)/m$

$$\sigma_x^2(s) = \frac{\sum_{\ell} (x_{\ell}(s) - \bar{x}(s))^2}{m}$$

we come to the expression

$$\sigma_x^2(s) = \frac{X_2(s)}{m} - \bar{x}^2(s).$$

In fact, to minimize $X_2(s)$ tends to minimize $\sigma_x^2(s)$ therefore ensuring a uniform distribution of crossovers. It should also be noted that $X_1(s)$

and $X_2(s)$ are not necessarily correlated, suggesting that a more convenient approach might be the simultaneous optimization of both.

Breuer [Br77] defined another objective function based on the minimization of the largest value

$$Y(s) = \max_{\ell} \{x_\ell(s)\} .$$

Following Steinberg's result [St61] we may consider this objective equivalent to a $X_k(s)$ with k of a sufficiently high order.

10. CONCLUDING REMARKS

The placement problem was here defined from a theoretical point of view and an attempt was made in order to systematize its most widely accepted figures of merit. For this effect a number of objective functions was defined, which are independent of any particular technology or routing scheme.

It should however be pointed out that the optimization of any of the defined objective functions leads to a complex combinatorial problem. An example is the case of $M_1(s)$, and of $X_1(s)$ as well, leading to the quadratic assignment problem which is known to be NP-complete. On the other hand, a careful choice must be made among those objective functions, since the optimal solutions they produce may be conflicting. This is the case of a solution obtained strictly on minimal total distance $M_1(s)$ that may give poor results in terms of even distribution of crossovers $X_2(s)$.

From the above discussion it is clear that a convenient criterion for the placement problem should involve a combination of two or more of the referred objective functions. Moreover, since the problem cannot be "solved" in the mathematical sense, it is admissible that approximation algorithms be designed in terms of such combinations.

The simultaneous optimization of objectives may also bring other desirable properties as side-effects. For example, to minimize $M_2(s)$ together with $X_2(s)$ will tend to distribute the density of from-to pairs evenly on the board.

It is our belief minimal total distance $M_1(s)$ balanced by an even distribution of crossovers $X_2(s)$ is, for most of the cases, the most convenient combination of objectives. We think that efficient algorithms can be designed as an attempt to minimize both functions, in such a way that $X_2(s)$ is favoured in the cases where conflict arises. As a result, a near minimal total distance would be achieved and potential areas of wiring congestion avoided.

CHAPTER II

PLACEMENT METHODS

1. INTRODUCTION

This chapter is intended as a guide and motivation for the following ones, where a new placement method will be fully described and discussed in mathematical terms.

The placement problem as defined previously has been studied under different approaches,depending on the technology in use and the choice of the objective functions to be optimized.

In the present Chapter we specify three basic approaches and accordingly we classify the existing placement techniques.

Our new method is then globally described and classified. Its scope and restrictions are considered and associated problems are raised.Special concepts like "knot" and "connectivity tree" treated later in detail, are sketched here.

The final section was designed as a summary of the subsequent Chapters where the method will be fully analysed.

2. CLASSICAL METHODS

A variety of placement techniques has been developed which directly aim to optimize one of the classical objectives: minimal total distance and even distribution of wires over the board surface.

In general, over an initial configuration either random or constructive, an iterative scheme swaps pairs (or groups) of components while some improvement is observed. This approach makes this class of methods particularly suitable for regularly structured environments with fixed module locations.

2.1. Initial placement

It has been debated in the literature whether it is better to use a random start or a constructive-initial placement as an initial solution to the iterative improvement algorithms. Proponents of the former argue that the computation time to generate a constructive-initial placement is better spent generating several random starts from which the best solution can be chosen.

However, experimental work on the subject [HW76] shows that there is no correlation between the total distance of a random initial solution and the final total distance after placement improvement. Therefore, the best initial solution does not necessarily lead to the best final placement so that each random start needs to be followed by the iterative improvement algorithm.
It is also stated by Hanan, Wolff and Agule that iterative-improvement algorithms tend to run longer when starting with a random placement as compared with starting with a good constructive-initial placement. Detailed experimentation showed that, generating k random starts and following each one with iterative improvement takes at least k times as long as generating one constructive-initial placement and following it with a placement-improvement

algorithm.

Hanan and Kurtzberg [HK72] present a general discussion of the class of constructive-initial placement techniques and a description of the most common algorithms. In this class of methods, modules are selected one at a time and positioned in the partially formed configuration. The particular rules for *selection* and *positioning* of the modules, define the specific methods.

Once an acceptable initial configuration is obtained an iterative scheme is then applied, depending on the objective function adopted.

2.2. Minimal distance methods

In this class we include all iterative placement methods that aim to minimize the total distance $M_1(s)$, therefore reducing the placement problem to the quadratic assignment problem.

Over the last two decades a large number of papers on this subject has appeared in the literature. Hanan and Kurtzberg [HK72] give a detailed description of the most important minimal distance placement techniques and a list of references. Subsequent to this work,Hanan,Wolff and Agule [HW76] performed a comprehensive study of the relative efficiency of those techniques. Detailed experimentation showed that, for large problems, the only viable combination of placement algorithms is based on the *force-directed pairwise relaxation* algorithm. It not only achieves the best placement but it also achieves it in the shortest computer time.

Pairwise interchange algorithms test all $n(n-1)/2$ possible pairs of n modules for interchange.Whenever one trial interchange results in a reduction of $M_1(s)$ this interchange is accepted, otherwise the modules return to their previous positions. In the force-directed algorithms, a force vector $\bar{F}_M$ is computed for each module M

$$\bar{F}_M = \sum_i c_{Mi}\ \bar{s}_{Mi}$$

where $\bar{s}_{Mi}$ is the vector distance from M to i. This force vector allows the target point, where the sum of forces on module M is zero, to be computed. The target location of M is then chosen within a certain neighbourhood ε_M of its target point.

The force-directed pairwise relaxation algorithm is a combination of both concepts. A module A is chosen for a trial interchange with a module B located in ε_A, only if A is located in ε_B. This procedure clearly limits the number of trial interchanges and since two modules are optimized at a time, computation time is reduced considerably.

The resulting solution, obtained strictly on the associated quadratic assignment problem, may then be adjusted to the original placement problem. This is still accomplished by means of the force-directed pairwise relaxation algorithm adapted in order to minimize the minimum spanning tree distances of the placement problem.

As already pointed out in Chapter I, minimal distance methods produce solutions where heavily connected modules tend to be clustered together. On the other hand, pairwise interchange algorithms perform badly whenever the sizes of modules are significantly different. The first problem can be avoided by manual intervention, the latter problem can be solved by subdividing a large module into its constituents.

2.3. Analytical methods

The analytical methods use mathematical techniques to directly optimize a defined placement objective. The resulting solution is then used to produce the placement configuration.

Force-directed placement algorithms [CM73] [QB79] are classical representatives of this approach. A set of simultaneous equations is established, which models a system of attractive forces between interconnected components and repulsive forces between unconnected components. The solution to this system gives the equilibrium position of each module. This solution must then be adapted to the particular type of environment: in regular structured boards components are assigned to the slots and in the general case components are spread out in order to eliminate overlaps. Quinn and Breuer [QB79] present a mathematical formulation of the subject and a technique for solving the set of equations.

More recently, another class of analytical methods is being developed where the objective function is basically the total weighted squared distance $M_2(s)$. Cheng and Kuh [CK84] formulate this approach to the placement problem in analogy with resistive network optimization. A method is proposed for solving the optimization problem and for using this solution to place the modules on regularly structured boards. With a similar formulation, Blanks [Bl85] evaluates lower bounds for $M_2(s)$ and uses these bounds in the development of a new pairwise interchange placement technique.

A mathematical model can also be constructed for the general placement case. In the method presented by Markow et al. [MJ84], the objective function depends on the total rectilinear distance and on a criterion of the chip size. That function is also subject to a number of restrictions which define a position for each block relative to the other blocks and the chip boundaries. The used iterative optimization algorithm gives, at each step, a feasible solution and an estimation of the distance from this point to the optimal solution. The method due to Sha and Dutton [SD85] follows a similar approach. However, the objective function is defined in terms of a squared Euclidean metric and there are no integer variables,which enhables the use of a classical optimization procedure.

2.4. Even distribution methods

This class includes all iterative placement methods that are intended to minimize the number of crossings over a set of boundary lines.

At each cut line, the problem of grouping components into two subsets so as to minimize the interconnections is typically a partitioning problem. This subject will be discussed in Chapter III of this thesis.

The partitioning algorithm required by the placement methods included in this class is usually based on Kernighan and Lin's procedure [KL70][SK72]. This procedure is essentially an iterative interchange between groups of modules selected from both subsets.

The location of boundary lines and the order in which they are processed characterize the different placement methods. Breuer [Br77] introduced the concept of sequential optimization of cut lines and elaborated a thorough discussion on the subject. The placement algorithms of the *min-cut* class, due to Breuer, are particularly suited to regularly structured boards with fixed module locations. Two min-cut procedures, namely *quadrature* and *slice/bisection*, specify the position and sequence of cut lines. In the quadrature algorithm the board is repeatedly bisected by alternating vertical and horizontal lines. By first processing cut lines in the centre of the carrier, interconnections are pushed away from this region. This fact makes quadrature suitable for boards with a high density of routing in their centre. Slice/ /bisection is best suited to boards where there is a high interconnect density at the terminals. The board is divided into successive rows and each row is in its turn repeatedly bisected.

In other min-cut algorithms, like the one developed by Corrigan [Co79], the order of partitioning is not predetermined. A set of "directives" defined by the user during the process execution, is made to meet the particular characteristics of the assembly.

Although the placement methods included in this class have been originally designed for regularly structured boards,they introduced a number of concepts that can be applied to other types of environment.The basic terms cut line and partition are often present in hierarchical placement techniques dealing with the general case.

3. HIERARCHICAL APPROACH

As discussed in Chapter I of this thesis, the increasing complexity of integrated circuits with cells of dissimilar shape and size, brought up the need for hierarchical layout methods. With the advent of VLSI,hierarchical placement methods started to be considered.

There are basically two ways to handle the placement of general cells under a structured approach. Either the whole circuit is successively partitioned until the cell level is reached or, starting with isolated cells, blocks of increasing complexity are built up.

3.1. Top-down placement

Lauther's min-cut placement algorithm for general assemblies [La79] is a typical representative of this approach. Starting with the total cell area and the desired shape of the final assembly, it is possible to calculate the expected dimensions of the substrate. The initial rectangle representing the whole board is divided into two rectangles of sizes adjusted to fit the total cell area in both subsets. The process is recursively applied to the resulting rectangles, until each subset contains a single cell.

For partitioning, a modified version of Kernighan and Lin's procedure is employed. The direction of cut lines is switched between horizontal and vertical for each iteration unless, at the last steps, cells do not fit into the available area.

Once each cell is allocated to a rectangular domain, its exact coordinates and dimensions must still be determined. The use of three post--processors, namely "rotation", "squeezing" and "reflecting", can still improve the final layout.

As pointed out by Wipfler et al. [WW82] the disadvantages of this method are as follows: Due to its sequential nature the min-cut procedure minimizes the number of crossovers locally but not globally. The results of the bipartitioning algorithm also depend on the initial configuration adopted. Moreover, the additional constraints concerning the size of the blocks (total area of all elements assigned to the block) are difficult to implement. In practice, this causes overlaps of cells.

In order to solve the first two problems, Wipfler et al. proposed a method based on the force-directed concept. A force-directed placement algorithm calculates the relative cell positions (circuit topology) and a cut algorithm transforms this relative placement into a geometric cell arrangement considering cell area. The cut algorithm is not a partitioning process, in the sense we have been considering, since it only concerns board areas and does not alter the circuit topology. In order to eliminate overlaps, cells can be "moved", "rotated" and "mirrored" in interactive mode.

We also include in the present category the type of placement methods that, while performing the space division phase in a top-down fashion, execute the partitioning of cells in a bottom-up mode. A clustering algorithm combines blocks into groups of high connectivity from the cell level towards the whole circuit, thus producing a *cluster tree* [MT79] [Ha82]. This tree structure is traversed top-down during the process of space division, so that each block is allocated to a rectangular area.

At each hierarchical level of a top-down placement, the space division phase can be interpreted as a Floor Plans problem [Ga81], where a combination of rectangular space costs and communication costs must be minimized. In order to simplify this question, most of the existing methods are of a binary nature so that space division is,at each step, limited to the bipartition of a rectangle. The problem can nevertheless be approached by means of Linear Programming techniques, as in [MM82] [HS82], permitting an effective optimization of the available area.

3.2. Bottom-up placement

As opposed to the former category, bottom-up placement methods start from one cell and add other cells one by one until all cells in a block are placed. Blocks are then combined in a similar manner and the process is repeated until the complete design is placed.

Let us examine the placement method due to Preas and van Cleemput [PV79], as a representative of this class. The block structure (partition of cells) is defined on functionality and a limit is imposed on the number of blocks which are to be processed simultaneously. For this reason a branch--and-bound technique can be used to examine all possible solutions. At a given level, whenever a new block is to be added to the partially formed configuration, every possible relative position has to be considered (i.e. above, under, to the left and to the right), as well as all possible rotations and reflections of the new block. The cost function to be minimized is the total area occupied by the blocks and their interconnections, with possible constraints on the maximum height, width or aspect ratio of the final assembly.

A bottom-up strategy can also be efficiently employed is standard cell and polycell assemblies. Richard [Ri84] uses a clustering algorithm to successively combine pairs of strongly interconnected (groups of) components. The binary tree structure thus generated is then used to determine the order in which the components are sequentially selected an placed.

For both approaches it holds that it is not enough to build a dense placement of the cells, without including some expected routing area. If this area was not included during the process, the cells would have to be subsequently expanded, thus producing an irregularly shaped assembly.

The expected routing area can be more accurately evaluated under a bottom-up methodology. It is also possible to perform, at each stage, a complete routing of the partial configuration. A placement method proposed by Malladi et al. [MS81] includes an estimation of the interconnection channel widths, as well as the number of interconnection crossovers and connection lengths. The algorithm due to Chandrasekhar and Breuer [CB82] deals with the particular case of a binary block hierarchy. At a given level, two rectangular blocks are combined in order to produce a new rectangular block in such a way that the total layout area including the interconnect space is minimized. In [BK83] a statistical model is used for estimating the necessary routing area at each hierarchical level and accordingly modify the initial placement solution, obtained by a force-directed algorithm.

All the previous classes of placement methods explicitly aim to optimize at least one of the classical objectives: minimal distance or even distribution. Bottom-up algorithms are apparently only intended to minimize the chip area. However connections do take space,and a global overview of the circuit connectivity is essential to an effective optimization. For that reason, the placement methods of the present category must rely on a starting solution, usually obtained with the aid of one of the classical algorithms.

It should also be pointed out that, to pack a given set of rectangular blocks into as small an area as possible is by itself a very complex combinatorial problem. Two-dimensional packing problems are known to be NP-complete and the search for suitable approximation algorithms has been a field of active research [GJ81].

4. THE USE OF GRAPHS

As discussed in Chapter I of this thesis, graph theoretic structures offer a convenient representation for some aspects of the circuit layout problem. In the representation of circuits,they can model a large range of circuit properties. Graphs are also present in the form of trees, when representing different types of layout hierarchies.

In this section we include all placement methods that explicitly make use of graph models.

4.1. Topological approach

Topological layout methods, rather than executing the usual phases of placement and routing, first construct a graph model for the circuit. This graph represents the topological aspects of the circuit as faithfully as possible, while neglecting all geometrical information. The graph is then embedded in one or more planes with the restriction that no two edges must intersect except at the vertices. The final step consists of transforming the topological layout into a physical layout that takes into account the geometrical properties.

The topological approach has the property of objectively considering circuit planarity as a factor of circuit layout. Under the usual methodology the problem is only taken into account during the routing phase, when components have already been assigned to their definitive positions. In the case of PC boards two distinct connections are not allowed to cross, which causes a number of plated-through holes or vias. The technology of integrated circuits may provide a form of "cross-overs", which however consume some physical area.

There already exist efficient, linear-time algorithms for testing whether a graph is planar [HT74]. Yet, to embed a given graph in a planar surface so as to minimize the number of edge-crossings, has proved to be another NP-complete problem [GJ83].

Van Cleemput elaborated a comprehensive study of topological layout methods [Va76]. A very concise mathematical formulation of graph models was established and a number of algorithms, dealing with the graph embedding problem, were presented. A variety of models and embedding algorithms has been developed, which are able to capture the particular topological properties of different technologies [NJ85].

The space partition method reported in [MM82] [HS82] makes use of the dual of a planar graph concept in a different approach to circuit embedding. An important property of a planar graph is that it has a planar dual. If certain conditions are satisfied, a planar original graph (POG) will generate a rectangular dual graph (RDG). Figure 3 shows an example of this correlation.

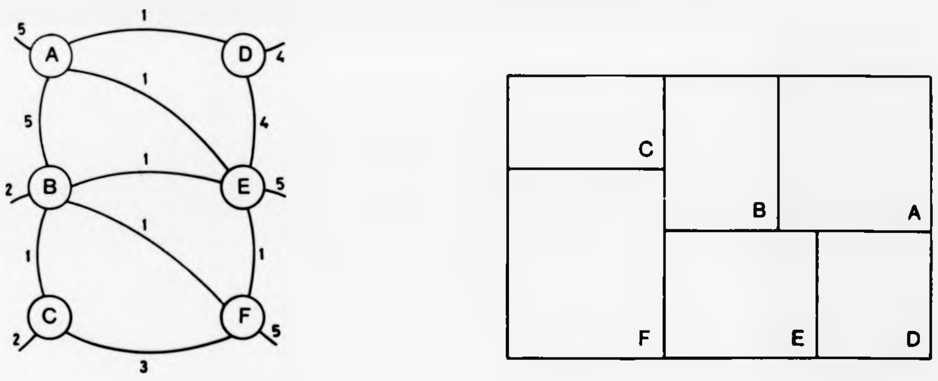


FIG. 3 a) Planar original graph b) rectangular dual graph

Rectangular areas in the RDG represent nodes in the POG and contiguity between rectangles in the RDG represents connectivity between the corresponding nodes in the POG. The dimensions of the rectangles are evaluated as a function of the connectivity between blocks, the cell count for each block and particular technological constraints. A search for the solution with minimum or near minimum total area is then performed by means of Linear Programming techniques.

4.2. Polar graph concept

The idea of representing an arrangement of non-overlapping rectangles by a polar graph had already been applied to the layout of integrated circuits. However, only with the advent of hierarchical placement methods, was this concept fully explored. Figure 4 shows a sequence of stages during a top-down placement process [La79].

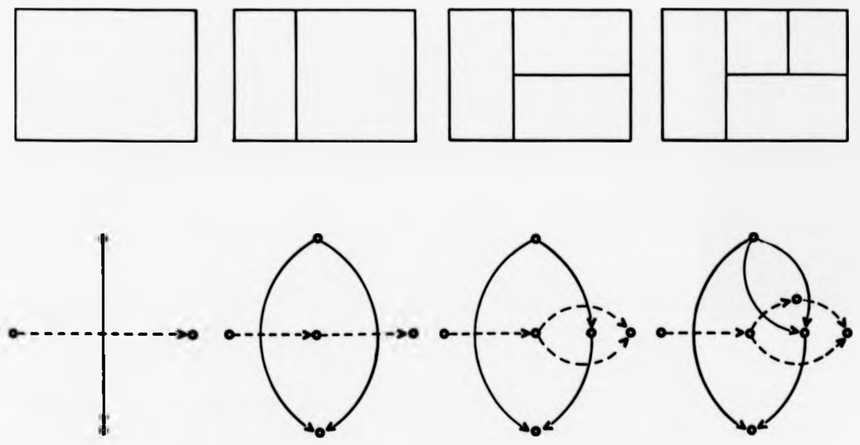


FIG. 4 - Polar graphs representing top-down placement

At each stage, the arrangement of blocks is represented by a pair of planar, acyclic directed graphs. The graph G_x represents the x-dimension and is denote by dashed lines whereas the y-dimension graph G_y is denoted in full.

There is a one to one correspondence between the edges of G_x and G_y. When the algorithm starts, the two graphs contain one edge each. This pair of edges represents the initial rectangle, whose dimensions are evaluated in terms of the total cell area.

The partitioning of a set of cells is represented by a splitting of the corresponding edge pair, into two new edge pairs. The lengths of the new edges are adjusted, according to the total cell area in each subset.

At the final stage, each edge pair represents the real dimensions of a single cell. Note that until now we only have a relative positioning of the cells, as represented by the edges in both graphs. To find the actual coordinates of the cells, a longest path algorithm must still be applied.

This polar graph representation of blocks has also been employed by bottom-up placement algorithms [PV79]. The model is similar to the one already described, with the difference that an edge pair always represents a single cell instead of entire blocks.

From the basic model generated during the placement phase, a more detailed version can be derived, representing the positions and widths of routing channels [PG78][La79][Ha82].

4.3. Symbolic layout

A different approach to the layout problem involves the input, by the circuit designer, of a diagram representing the circuit topology [Wi78]. A set of programs adjust this diagram in order to include all significant design rules and minimum geometry constraints. This is followed by the use of an automatic compaction routine.

Under this semiautomatic approach, more sophisticated systems may provide the access to a collection of routines to improve the different stages of the process [Hs81][LJ84][RR85].

4.4. Compaction

An increasing number of layout automation systems, dealing with custom IC design, rely on some form of compacting layout configurations into as small an area as possible. Compaction is a relatively new field in CAD [AG70], which is rapidly becoming a vital tool in the design of custom IC layouts. To this subject, an interesting parallel has been suggested by Cho [Ch85] between IC layout and computer programming: "doing layout with compaction is like programming in high-level languages... As projects become more complex and compaction improves, doing layout with compaction may dominate the custom design process, just as most programming projects today are done with high--level languages".

Two-dimensional compaction is a complex problem, which has proved to be yet another NP-complete problem in layout automation [AH74]. For that reason, most of the proposed methods separate the compaction problem into two independent processes: the x-compaction, during which elements can only move horizontally to the left and the y-compaction, where elements move strictly vertically to the bottom.

Several different approaches have been proposed for solving the compaction problem [Ch85]. Here we will outline two main classes of methods, namelly *constraint graph* and *compression ridge*.

Under the first approach, a graph structure is first built to indicate the relative positions and the minimum distances required among the elements. Similar to the polar graph concept, a constraint graph may however be made to represent arbitrary rectilinear areas [HW84]. The x-compaction is performed with basis on the graph G_x (as defined in § 4.2) and the y-compaction is performed over G_y, in two independent processes. In a second phase of the method, both graphs are solved using a longest path technique.

Finding a shortest path in a directed graph is a well-known computer science and operational research problem. Changing from shortest path to longest path involves only minor modifications to the chosen algorithm [GK85]. Working on the graph G_x, the minimum required x-dimension of the chip can be evaluated as the length of the longest path in G_x. The x-coordinate of each component must be, at least, the length of its own longest path away from the source node. A parallel process evaluates the y-coordinates of components.

The basic constraint graph approach can be improved in a number of directions [Ch85]. In particular, compaction methods are being developed which are compatible with hierarchically designed layouts. A further minimization of the total chip area might still be achieved by rotating particular components along the critical path, whenever this operation does not result in a noticeable increase in the opposite chip dimension. This approach could require several iterations and repeated longest path evaluations but appears viable by means of network programming techniques.

Compression ridge methods [AG70] perform compaction by cutting off empty spaces. Each x-compaction process cuts off an equal-width vertical segment (not necessarily continuous) that forms a complete column. X-compaction and y-compaction are alternated until no complete column or line can be found.

Compaction was originally introduced as a means of packing rough sketches or symbolic diagrams to produce IC layouts. It may however become a powerful tool when used as part of a fully automated layout system. Placement and routing may then be performed on a larger virtual area and subsequently compacted in order to produce the final layout [RM83].

5. A NEW PHILOSOPHY

The intrinsic complexity of the placement problem together with the diversity of layout environments, has led to the development of a variety of placement techniques. Depending on the technology in use and the choice of objectives, different approaches have been considered and a number of placement concepts has been originated.

The layout of integrated circuits motivated the use of hierarchical methods as a means of dealing with increasing circuit complexity. However this approach is not, in our belief, completely explored. Few attempts have been made to build up the hierarchical structure and then to map it into the board surface.

In most of the top-down placement methods, the hierarchy is defined through a sequence of partitions that locally minimize the number of interconnections. As already pointed out, this technique requires a good initial solution and the interchange of blocks is constrained by their relative sizes. Moreover, the iterative nature of the partitioning process is not actually defined in mathematical terms. Blocks are successively interchanged as long as some improvement is found, but there is neither a guarantee of the convergence of the process nor a limit on the number of iterations.

Bottom-up placement methods also rely on a starting solution for a global overview of the circuit connectivity. The hierarchy of blocks is defined by the user and based on functionality.

We suggest that a more effective approach consists of elaborating first the entire circuit hierarchy and then embedding it in the board. We believe that a tree structure can be built, modelling the hierarchy of blocks, in such a way that desirable properties of the final placement are incorporated. This tree structure can subsequently be mapped into the circuit plane in accordance with the particular type of environment.

A major advantage of the proposed approach lies in the fact that placement objectives are included in the building of the structure. In particular, a hierarchy can be elaborated in terms of the classical objectives, i.e. minimal total distance and even distribution of connections.We assume that efficient algorithms can be designed for the building of such a structure taking account of a chosen set of topological and geometric circuit properties.

Under this approach, no initial configuration is required and no iterative interchange is performed either during the building or the embedding of the tree structure. This structure represents the relative positions of the blocks on the board, which were defined in terms of circuit properties and placement objectives. It can only be altered by manual intervention or by geometrical board constraints.

The hierarchical structure at each level consists of detailed information about the elements included in each block as well as their connectivity. This information may became a valuable legacy to a subsequent hierarchical router.

In the next paragraph a new placement method, designed in accordance with the proposed philosophy, is globally described. It combines the expedience of a hierarchical approach with the global circuit overview provided by classical methods as well as making extensive use of graph theoretic models.

6. PROPOSED METHOD

Consider a physical circuit (C,N), defined by a set C of components interconnected by a set N of nets and also a circuit board. This system may represent any type of placement environment such as a regularly structured PC board or a general cell VLSI chip.

The initial phase of the placement method proposed consists of representing (C,N) by a suitable *graph model*. As discussed in Ch. I - §8, a variety of graph theoretic structures provide a broad range of choice for the most adequate model. In the implementation of the method, the classical "component-to-node, connection-to-edge" representation was adopted. It is a simple model for the basic topological circuit properties, which also provides a convenient representation for other types of design requirements. Artificial weights may, at this stage, be imposed on specific edges in order to reinforce vital connections or to favour a required proximity between components.

The graph model acts as a basis for the building of a tree structure representing the circuit hierarchy as suggested in the last paragraph. A binary hierarchy was adopted in the implementation of the method, in order to simplify the following space division phase.

The tree structure is therefore a binary tree, called a *connectivity tree*, where each leaf represents a graph node and the root represents the whole graph. Figure 5 shows a connectivity tree for the sample graph model indicated in Ch. I. § 8.

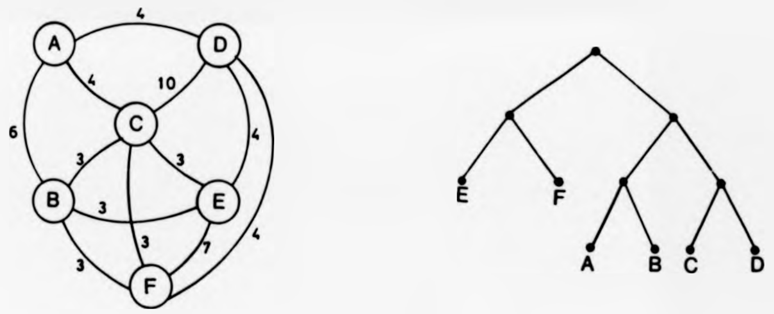


FIG. 5 - The connectivity tree concept

The way in which graph nodes are associated is determined by the choice of placement objectives to be optimized. As discussed in Ch. I. § 10., we assume that minimal total distance balanced by an even distribution of connections is, for most of the cases, a convenient combination of objectives. In the following Chapter we present a study of the definition of the connectivity tree and also a tree building algorithm based on the optimization of such objectives. This algorithm works in a bottom-up fashion, by successively associating pairs of nodes. At each tree vertex the resulting coalescence of nodes, or pairs of nodes, will be named a *knot*. This concept is intended to depict a group of interconnected nodes, with all the internal and external incident edges.

At this stage, the user may be given access to the tree structure generated by the algorithm and eventually alter the distribution of artificial weights on the edges of the graph model in order do obtain some desired effect.

In the next phase of the method, the connectivity tree is mapped into a given board area. Specific characteristics of the particular type of environment must now be taken into account. This problem is discussed in Chapter IV of this work, where two algorithms are proposed, one for regularly structured boards with fixed slot positions and the other for the general case of a continuous plane.

The *mapping process* is considerably simpler for the first type of board. In the general case, an intermediate stage is required in order to evaluate the expected routing area. Since the connectivity tree is built in a bottom-up order, this stage may however be associated with the tree building phase, in a similar manner to the one discussed in § 3.2 of the present Chapter. Each tree vertex will therefore contain an estimation of the corresponding layout area including the interconnect space. The tree root will provide information about the expected total layout area of the assembly.

To embed a given hierarchical structure in a planar surface is essentially a space partitioning process. In the simple case of a binary tree structure this process is, at each step, reduced to the subdivision of a rectangle into two rectangles of specified areas. Figure 6 describes a sequence of stages in the embedding of the sample connectivity tree depicted in Fig. 5.

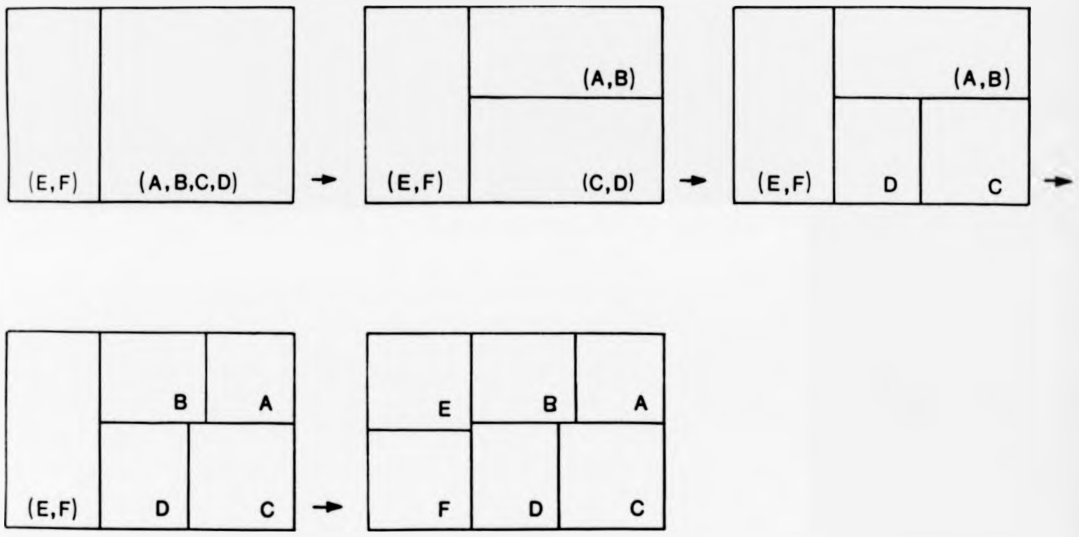


FIG. 6 - The space partition phase

Each one of the basic circuit components is now allocated to the centre point of a singular rectangle. Local adjustments, of each component within its rectangular domain, may still be performed in order to enhance the routability of the placement solution.

7. SCOPE AND RESTRICTIONS

A new placement method has been proposed as a representative of a fully hierarchical placement philosophy. In the present paragraph we investigate the correlation of an approach with other stages of the layout process and point out the main difficulties encountered in the implementation of the method.

The proposed placement approach is compatible with a wide range of layout environments. A collection of algorithms will map the connectivity tree into different board geometries.This fact enables the use of the method at various layout levels of the system hierarchy discussed in Ch. I. § 5. The connectivity tree of a given circuit then becomes a branch of the tree structure representing the whole physical hierarchy of the system. Under this perspective, we believe it is viable to apply similar algorithms to the one used for the building of the connectivity tree to other aspects of the layout problem. The partitioning of the circuit into several boards so as to minimize interboard connections and the grouping of gates into components with minimal intermodule connections, are two likely candidates.

Subsequent to the placement of a PC board, the step of gate assignment is essentially a placement at the component level. Wire-list determination and pin assignment can be performed in terms of the placement solution.

The connectivity tree also provides valuable information to be used during the routing phase. We suggest that the sequence in which the wires are processed, could be defined in terms of a bottom-up tree ordering. Since one major objective in the building of the connectivity tree is minimal total distance, components associated at the lower levels and therefore going to be placed together, are also the most strongly interconnected ones. A number of hierarchical routing methods has been proposed [LS80][BP83][RM83] , where a bottom-up scheme defines "saturated zones" on the board that are successively processed in a shortest-first order. Our approach can be made compatible with such a routing scheme if "saturated zones" are identified with the board areas defined by the space partition phase.

With respect to the implementation of the proposed placement method, the main difficulties concern the space partitioning phase. These difficulties are originated by two major factors: specific design requirements and a high density of components.

Examples of design requirements are the location of edge connectors, the preplacement of components or blocks and the existence of preferential and forbidden board areas. The space partition algorithms are compatible with a limited number of preplaced elements because these will act as fixed points in the mapping of the tree structure. However, the existence of a large number of preconditions tends to conflict with the established placement objectives. Design requirements will then be satisfied to the detriment of the structure defined by the connectivity tree.

At the end of the tree building phase, we have an estimate of the minimum total area required for the placement of the circuit. In most cases, the available board area is larger than the required minimum and the mapping algorithm will distribute components uniformly on the board surface. Tree-embedding algorithms are designed in such a way that the area attributed to each block is proportional to the total area of the components it contains. The area occupied by components, and consequently the empty spaces, will then be evenly distributed over the available space.

In the case of very dense boards this proportionality is difficult to achieve, in particular when elements of very dissimilar size are involved.

The binary top-down approach to space partition adopted here may also cause fitting difficulties at the lower levels, in the cases of a high component density. Since the implemented version of the method did not include any selection of component orientation, this problem had to be solved by manual intervention. However, it appears feasible to consider a more elaborate version of the method, where the best relative orientation of two coalescing components is determined during the tree building phase.

A more general solution to the difficulties caused by very dense boards, in particular when a minimum total area is required, would comprise two stages. The circuit layout on a virtual area, estimated as sufficient for placement objectives to be accomplished, followed by the use of an automatic compaction routine.

8. ABSTRACT OF THE FOLLOWING CHAPTERS

In the following two Chapters the placement method proposed will be studied in detail. Chapter III comprises a discussion, in mathematical terms, of the definition and building of the connectivity tree. The tree--mapping process is described in Chapter IV. Two algorithms are presented, one for regularly structured boards and the other for the general case of a continuous plane, both taking into account the problems raised in the last paragraph. In Chapter V we report a summary of the results of the implementation of the proposed placement method.

CHAPTER III
GRAPHS, PARTITIONING AND TREES

1. INTRODUCTION

As seen at the end of the last Chapter, a concept fundamental to this work is a binary tree representing the connectivity structure of a given circuit.

In this Chapter we discuss the definition and building of the connectivity tree. Starting with some graph theoretic concepts, we analyse the partitioning problem and its relationship with placement. Tree-objectives are established and a tree-building algorithm is presented.

CHAPTER III

GRAPHS, PARTITIONING AND TREES

1. INTRODUCTION

As seen at the end of the last Chapter, a concept fundamental to this work is a binary tree representing the connectivity structure of a given circuit.

In this Chapter we discuss the definition and building of the connectivity tree. Starting with some graph theoretic concepts, we analyse the partitioning problem and its relationship with placement. Tree-objectives are established and a tree-building algorithm is presented.

2. NOTATION AND CONCEPTS

A *graph* is a system consisting of a finite non-empty set $\{x_i | i=1,2,...,n\}$ of *nodes* and a family of unordered pairs of distinct nodes (x_i,x_j) called *edges*. The node set of a graph G will be denoted by $N(G)$ and the edge set by $E(G)$.

With *all* pairs of nodes (x_i,x_j) a real function $c(x_i,x_j)$ called *weight* is associated, such that:

a) $\forall\ x_i,x_j \in N(G)\ :\ c(x_i,x_j) \geq 0$.

b) $c(x_i,x_j) > 0$ if and only if $(x_i,x_j) \in E(G)$ i.e. stricly positive when there is an edge between x_i and x_j. Note that $c(x_i,x_i) = 0$ for all x_i.

c) $c(x_i,x_j) = c(x_j,x_i)$.

Given a graph G, a *knot* A is a subset of G consisting of a set of nodes $N(A) \subset N(G)$ and all edges incident to them, with the initial weights. Considering the graph shown in figure 7, A is the knot built upon the nodes x_3, x_4 and x_5.

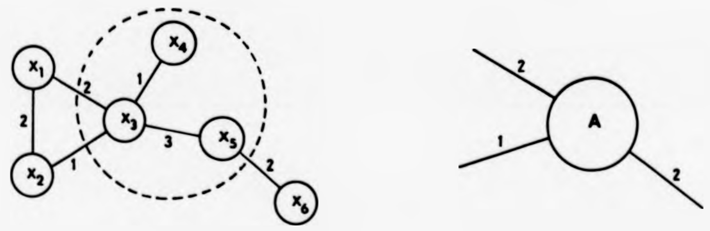


FIG. 7 - The concept of knot

Edges between nodes in A are called *internal* to A and not represented in the diagram. Similarly, the edges connecting nodes in A to the outside are said to be *external* to knot A. In the simplest case, a knot may consist of a single node $x_i \in N(G)$ with all edges incident to it.

We will now define the operation *coalescence* (o) between a pair (A,B) of knots with disjoint sets of nodes, i.e. $N(A) \cap N(B) = \phi$, as

$$\circ : (A,B) \mapsto A \circ B$$

The result $A \circ B$ is a new knot such that

$N(A \circ B) = N(A) \cup N(B)$

and $E(A \circ B) = E(A) \cup E(B)$, as illustrated in figure 8.

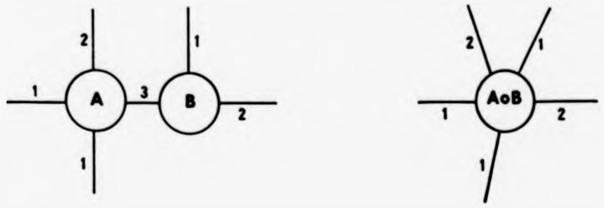


FIG. 8 - Coalescence between knots

The following concepts are made to introduce a measure of the edges in a knot, both the internal and the external ones.

Let us consider the definition of *degree* $\delta(x)$ of a node $x \in N(G)$:

$$\delta(x) = \sum_{x_i} c(x,x_i) = \sum_{x_i \neq x} c(x,x_i)$$

since $c(x,x) = 0$.

Similarly the degree of a knot A will be the sum of weights for all external edges, i.e.

$$\delta(A) = \sum_{x_i \in A} \sum_{x_j \notin A} c(x_i,x_j).$$

Between two knots A and B with disjoint sets of nodes, we determine their *interconnectivity* $\Psi(A,B)$ as

$$\Psi(A,B) = \sum_{x_i \in A} \sum_{x_j \in B} c(x_i,x_j)$$

and may therefore evaluate the resulting degree of two coalescent knots

$$\delta(A \circ B) = \delta(A) + \delta(B) - 2\Psi(A,B) \cdot$$

Considering now the internal edges in a knot, we define *intraconnectivity* $\Phi(A)$ of a knot A as the sum of weights of all internal edges, i.e.

$$\Phi(A) = \frac{1}{2} \sum_{x_i \in A} \sum_{x_j \in A} c(x_i,x_j).$$

Note the coefficient 1/2 is present because each edge would otherwise be counted twice.

In the trivial case of a knot consisting of a single node x, there are no internal edges and $\phi(x) = 0$; as to a coalescence A o B we determine:

$$\phi(A \circ B) = \phi(A) + \phi(B) + \Psi(A,B) .$$

For the total measure of both internal and external edges in a knot A we define *connectivity* $\sigma(A)$:

$$\sigma(A) = \delta(A) + 2\,\phi(A).$$

In the case of a single node x, connectivity is the same as degree, i.e.

$$\sigma(x) = \delta(x)$$

and for two coalescent knots

$$\begin{aligned}\sigma(A \circ B) &= \delta(A \circ B) + 2\,\phi(A \circ B) = \\ &= \delta(A) + \delta(B) - 2\,\Psi(A,B) + 2\,\phi(A) + 2\,\phi(B) + 2\,\Psi(A,B) = \\ &= \sigma(A) + \sigma(B)\end{aligned}$$

which makes connectivity additive with respect to knot coalescence. Therefore we may compute the connectivity of a knot A in terms of the degrees of its nodes

$$\sigma(A) = \sum_{x_i \in A} \sigma(x_i) = \sum_{x_i \in A} \delta(x_i).$$

Regarding the whole graph G as a knot without external edges, i.e. $\delta(G) = 0$, we have

$$\sigma(G) = \delta(G) + 2\,\phi(G) = 2\,\phi(G)$$

and thus

$$2\phi(G) = \sum_{x_i \in G} \delta(x_i)$$

which is Euler's first theorem of graph theory.

Let $\eta(A)$ be the number of single nodes in a knot A; we determine *ratio of connectivity* $\rho(A)$ as the quotient

$$\rho(A) = \frac{\sigma(A)}{\eta(A)} .$$

Note that the ratio of connectivity of the whole graph G is

$$\rho(G) = \frac{\sigma(G)}{\eta(G)} = \frac{\sum_{i=1}^{n} \delta(x_i)}{n}$$

i.e. the average degree of its nodes.

3. THE PARTITIONING PROBLEM

A *tree* can be defined as a connected (every pair of nodes are joined by a path) graph, with no cycles (closed paths). In particular, a binary tree is characterised by a *root* of degree 2 and *leaves* of degree 1 whereas all other nodes are of degree 3. In this work nodes of trees will be called *vertices*, in order to be distinguished from nodes of the initial graph.

We give the name *connectivity tree* to a binary tree representing the sequence in which coalescences of knots in a given graph are made. Each leaf represents a single node and all other vertices the result of two coalescent knots, the root thus containing the whole graph.

Figure 9 shows an example of connectivity tree and its original graph.

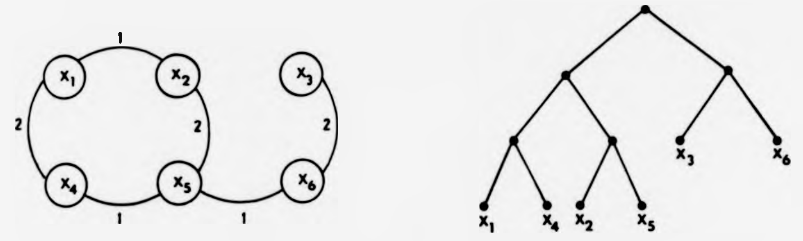


FIG. 9 - A connectitivity tree

Our aim is to find a special sequence of coalescences,that is, to build a tree satisfying certain predefined properties. We could, for instance, intend to maximize the interconnectivity at all coalescences made or, which is the same, minimize the degrees of all successive knots.

This is, in some sense, a bottom-up approach to the Graph Partitioning Problem, which can be stated as: given a graph G with costs (weights) on its edges, partition the nodes of G into subsets no larger than a *given maximum size*, so as to minimize the total cost (sum of weights) of all edges cut. Other formulations of the problem take different specifications into account such as, for instance, the number of permitted subsets. By traversing the tree top-down, each vertex may be regarded as a partition.

Graph partitioning is a NP-complete problem, which has been studied under several forms and in fields ranging from Combinatorial Analysis to Linear Programming. Due to its combinatorial complexity, a strictly exhaustive procedure to find a particular solution is completely infeasible. To see this let us count, in our case, the number of connectivity trees generated by a graph with n nodes. Let that number be denoted by τ_n.

Since coalescence is a commutative operation, there is only one tree for a graph with 2 nodes:

$$\tau_2 = 1$$

i.e., there is only one tree with two leaves.

Assuming we know τ_n let us now evaluate how many trees there are with n+1 leaves. It is known that a binary tree with n leaves has a total of 2n-1 vertices (n leaves plus n-1 internal vertices). The (n+1)th leaf can be coalescent to any of them thus, for each tree with n leaves there are 2n-1 trees with n+1 leaves, i.e.

$$\tau_{n+1} = (2\,n-1)\,\tau_n\,.$$

Similarly

$$\begin{aligned}\tau_n &= (2\,n-3)\,\tau_{n-1}\\ &= (2\,n-3)\,(2\,n-5)\,\tau_{n-2}\\ &= (2\,n-3)\,(2\,n-5)\,\ldots\,\tau_2\\ &= (2\,n-3)\,(2\,n-5)\,\ldots\,5.\,3.\,1\\ &= \prod_{i=1}^{n-1}(2\,i-1) = \frac{(2\,n-2)!}{2^{n-1}(n-1)!}\,.\end{aligned}$$

Note that this expression has a growth rate of a factorial times a power of 2, producing very large numbers for relatively low values of n, e.g.

$$\tau_{10} = 34459425 \;(\approx 3 \times 10^7).$$

Because it seems likely that any direct approach to find an optimal solution will require an inordinate amount of computations, heuristic methods are usually considered. They can produce good solutions (possibly even an optimal solution) and at the same time be sufficiently fast to be practical.

4. A LOOK AT PREVIOUS WORK ON PARTITIONING GRAPHS

The physical problem of dividing a circuit into two or more blocks with a minimum number of interconnecting wires is typically a partition in the mathematical sense. However, the explicit partitioning of graphs to enhance the performance of placement algorithms has received little attention. Usually partitioning is only considered as an interchange improvement process over an initial placement. This includes the min-cut methods [Br77] [Co79] [La79] all based on Kernighan and Lin's procedure [KL70] [SK72] adapted to obey particular constraints.

Basically the pairwise interchange strategy is as follows: "An element in A which is more strongly connected to elements in B than in A is considered a good candidate for a move to B. Similarly, elements in B are inspected for A candidates. The candidates from A and B are paired and the (hopefully positive) improvement from a simultaneous exchange is calculated for each pair; then the best pair is exchanged and the algorithm repeats until no further improvement can be found".

Kernighan and Lin's approach is to find, not just single pairs of elements to exchange, but entire groups of equal size. Therefore, even if the exchanging of subsets results in a loss, the exchange of whole groups may still give a profit. This is a more powerful approach but it still requires an initial placement and there is no way to tell how many iterations will be performed, each of them a partition problem on its own. Besides, the order of the partitions, their direction and position can all influence placement optimization. The order of partitioning needs careful selection since, for each element, the number of choices decreases with the continuation of the process.

Another partitioning method is the Ford and Fulkerson Max Flow - - Min Cut algorithm for networks [FF62]. The main difficulty in its use on the placement problem is that, on finding minimal-cost cuts it isolates nodes with only one low weight edge. As discussed in the next paragraph, a connectivity tree whose branches are too unbalanced in size should be avoided since it generates space fitting situations. Ford and Fulkerson's procedure makes no provision for constraining the sizes of the resultant subsets, and there seems to be no obvious way to extend it to include this.

Clustering methods [LL69] are of a much more intuitive nature on "identifying natural clusters" of nodes which are strongly connected in some sense. However, provision is not made for elements not obviously belonging to any set.

In our opinion, merging (bottom-up) processes seem a better approach to the partitioning problem than the splitting (top-down) ones. The main reason is a considerable reduction in the problem's complexity since, each level comprises only the selection of a few elements to be merged. Also, each resulting cost can be computed in a time independent of the total number of elements.

Important theoretical work was done by Luccio and Sami [LS69] giving some useful heuristic ideas which can be exploited to give good merging solutions. Given a graph, a group $\bar{A}$ of nodes is defined as *minimal* when, for any proper subset $A \subset \bar{A}$ where $A \neq \phi$, $\delta(A) > \delta(\bar{A})$. The main theorem of this paper establishes an interesting property, namely, that minimal groups cannot partially overlap, i.e., either

$$A \supset B \quad \text{or} \quad B \supset A \quad \text{or} \quad A \cap B = \phi.$$

A procedure based on this property is then outlined for determining all the minimal groups in a given graph.

5. ESTABLISHING OBJECTIVES

The concept of a connectivity tree, being the topological representation of a given network should, in its structure, approach the objective functions already established for the placement problem. In the present paragraph, those objectives are transferred to defining an optimal connectivity tree.

5.1. **Minimal total distance**

The basic aim is to keep together components (or groups of components) which have strongest interconnectivity. The operation coalescence results in the shortening of a distance between knots A and B. If most of the external edges in A go to B and conversely, then they should be coalesced.

The resulting A o B will have lower degree than either A or B.

This closely resembles Luccio and Sami's concept of minimal group. Translating to our own notation, we say a coalescence is *minimal* when

$$\delta(A \circ B) < \min \{\delta(A), \delta(B)\}.$$

In that case, since

$$\delta(A \circ B) = \delta(A) + \delta(B) - 2\,\Psi(A,B) < \delta(A)$$

we have $\delta(B) < 2\,\Psi(A,B)$ and similarly $\delta(A) < 2\,\Psi(A,B)$. Meaning that a minimal coalescence can only occur when

$$\Psi(A,B) > \max \{1/2\ \delta(A),\ 1/2\ \delta(B)\}$$

i.e., most of the external edges in both A and B are interconnections.

This is, of course, the ideal situation, not usual in practice. Yet, an efficient procedure for building a connectivity tree should be able to perform all existing minimal coalescences. Luccio and Sami's main theorem guarantees that each knot will at most have <u>one</u> possible minimal coalescence, which is easily inferred from the condition above.

Extending the concept to the whole tree we define an optimal connectivity tree, in the sense of a final placement with minimal total distance, as the one where all coalescences are minimal, and therefore the degrees of the vertices decrease upwards from leaves to root. Figure 10 shows an optimal min-dist tree, with the degrees at the vertices.

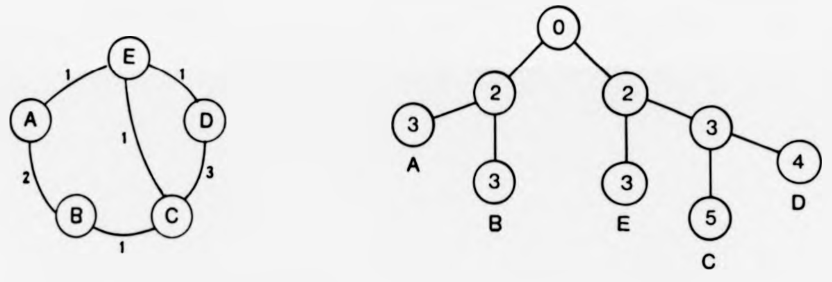


FIG. **10** - Optimal min-dist tree

5.2. Even distribution of wires

Another important measure of a good placement is the even distribution of wires across the board surface. In graph terms, the objective is to make the number of edges proportional to the number of nodes at each knot, i.e. a constant ratio of connectivity

$$\rho(A) = \frac{\sigma(A)}{\eta(A)}$$

for every knot A. At the next phase of the algorithm real components sizes will be taken into account and the proportionality extended to areas on the board.

In the sense of a constant ratio of connectivity, an optimal coalescence $A \circ B$ is the one where

$$\rho(A) = \rho(B).$$

Note that, in this case

$$\frac{\sigma(A)}{\eta(A)} = \frac{\sigma(B)}{\eta(B)} \quad \text{is also equal to}$$

$$\frac{\sigma(A) + \sigma(B)}{\eta(A) + \eta(B)} = \frac{\sigma(A \circ B)}{\eta(A \circ B)} = \rho(A \circ B)$$

i.e. the resulting knot still has the same ratio.

An optimal connectivity tree in the sense of even distribution is therefore the one where every coalescence is optimal and the ratio of connectivity constant for all vertices. Also, since the ratio at the root is the average degree of all nodes

$$\rho(\text{root}) = \frac{1}{n} \sum_{i=1}^{n} \delta(x_i),$$

the ratio at each vertex should be this value, or at least approach it. Fig. 11 shows a tree where the ratio is constant for all non-terminal vertices.

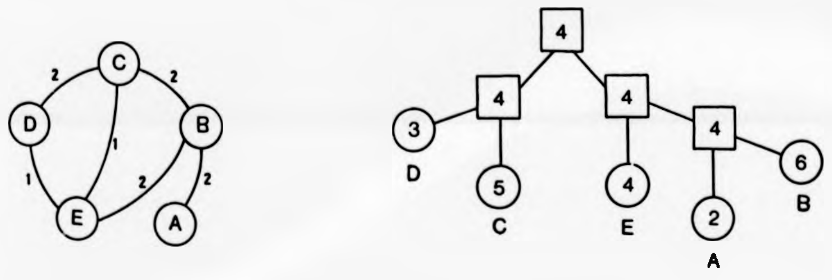


FIG. 11 - Optimal even-distribution tree

5.3. Balancing the tree

It is also desirable that the tree should not be too unbalanced. There are good practical reasons for that.

The first one is that the next procedure is a tree traversal, and the corresponding complexity can increase from $\mathcal{O}(n \log_2 n)$ to $\mathcal{O}(n^2)$ in the case of an extremely unbalanced one.

On the other hand, a large disproportion in the sizes of two coalescing knots, will be transmitted to the areas of the two rectangles to which they are going to be mapped. The smaller knot would be made to fit a long thin strip therefore increasing the distance between its elements. The extreme situation of an isolated component would leave a long empty wasted area at its side.

For these reasons it would be convenient to build a completely symmetrical connectivity tree, where at each vertex

$$\eta(A) = \eta(B).$$

It is interesting to note that this property can also affect the uniformity of the connectivity ratio. Since in this case

$$\rho(A \circ B) = \frac{\sigma(A) + \sigma(B)}{2\ \eta(A)} = \frac{\rho(A) + \rho(B)}{2}$$

then, the resulting ratio is the mean of the ratios of the coalescing knots.

5.4. Concluding remarks

Three objectives have been established for defining a connectivity tree leading to an optimal placement. Yet, for each of them, the optimal solution does not always exist and even a search for the best solution is by itself a complex combinatorial problem.

Moreover the three objectives are difficult to combine and are, in some cases, incompatible. For instance, a binary tree built strictly on min-dist is, as observed in practice, a very asymmetric one, since the low degree components tend to appear isolated at the upper branches.

Minimal total distance can also conflict with the constancy of connectivity ratio. Consider the situation on figure 12: fig. 12.a) shows the original graph, fig. 12.b) the tree built solely on min-dist and fig. 12.c) the tree aiming for a constant ratio. The degrees of the vertices are denoted in ○ and the ratios in □.

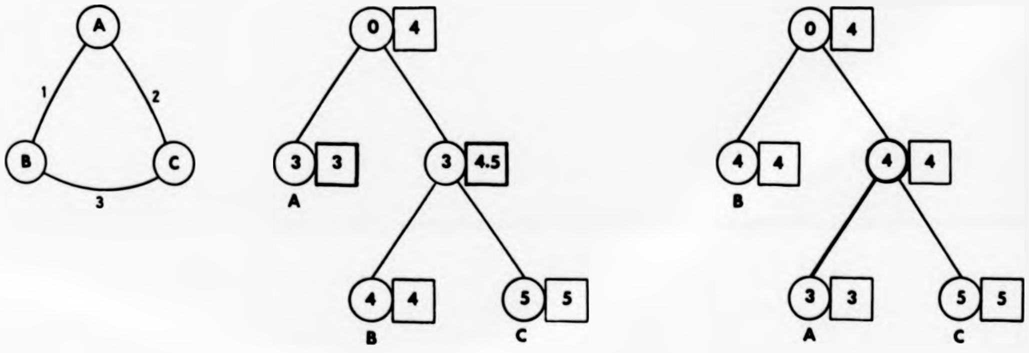


FIG. 12 - Connectivity trees of graph a), built on b) min-dist and c) even distribution.

5.4. Concluding remarks

Three objectives have been established for defining a connectivity tree leading to an optimal placement. Yet, for each of them, the optimal solution does not always exist and even a search for the best solution is by itself a complex combinatorial problem.

Moreover the three objectives are difficult to combine and are, in some cases, incompatible. For instance, a binary tree built strictly on min-dist is, as observed in practice, a very asymmetric one, since the low degree components tend to appear isolated at the upper branches.

Minimal total distance can also conflict with the constancy of connectivity ratio. Consider the situation on figure 12: fig. 12.a) shows the original graph, fig. 12.b) the tree built solely on min-dist and fig. 12.c) the tree aiming for a constant ratio. The degrees of the vertices are denoted in ○ and the ratios in □.

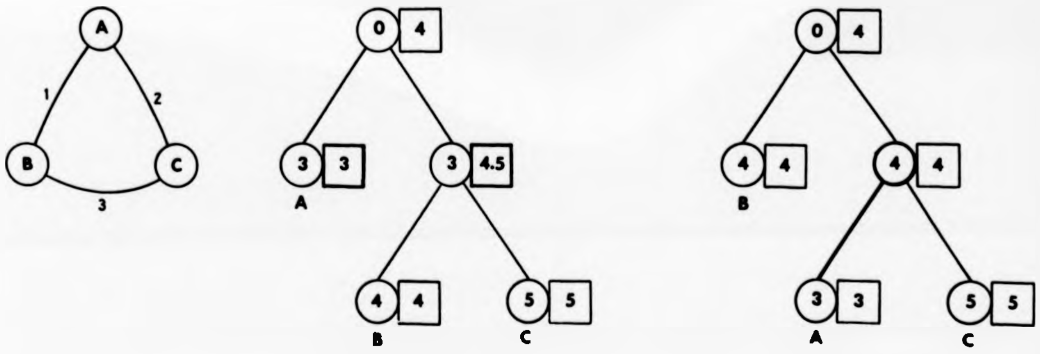


FIG. 12 - Connectivity trees of graph a), built on b) min-dist and c) even distribution.

In a conflict situation like this one the second solution should be chosen, favouring a final placement with even distribution of wires to a minimal total distance one, as discussed in Chapter I.

6. TREE BUILDING ALGORITHM

The remarks in the previous paragraph are, in a theoretical sense, a summary of the main difficulties in the placement problem. Hard, and sometimes contradictory objectives are contemplated.

The tree building procedure presented here tries to meet the three objectives as closely as possible, while being competitive in terms of running time and memory space. A merging (bottom-up) approach was followed as the simpler and more economical solution.

Let us recall the three objectives:

O_1 — Minimal total distance ($\delta(A \circ B) < \min \{\delta(A), \delta(B)\}$)

O_2 — Even distribution of edges ($\rho(A) = \rho(B)$)

O_3 — A symmetrical tree ($\eta(A) = \eta(B)$)

The algorithm starts by treating all nodes as knots and successively coalescing strategically chosen pairs of knots, until there is only one left (root of the tree).

At each step, a knot S is selected, which then determines the choice of C to be coalesced to. This choice of C is based on the interconnectivity $\Psi(S,C)$ and aims for O_1. The selection of S is intended to balance the tree both in size and ratio.

6.1. The selection of S

The way knot S is selected at each step is essential to objectives O_2 and O_3, to produce a tree balanced in terms of size as well as connectivity ratio.

The element going to lose its individuality is basically the smallest (S) in size, i.e. the one with the least number of nodes. By successively coalescing the smallest knot we obtain a more "fanned-out", i.e. symmetrical tree.

However, especially at the first steps working on isolated nodes, a selection based only on size is still undetermined, leaving plenty of scope for a secondary order to be introduced. From the equally smallest knots, the algorithm takes the one with lowest degree. The effect is similar to the former one: low degree elements coalesce first since they have less choice. They will tend to distribute evenly around the high degree ones. Experience showed that otherwise they remain isolated till the upper tree vertices.

6.2. The choice of C

This is mainly a search for a minimal coalescence with knot S. The algorithm looks for the "closest" (C) element to S in order to minimize their distance.

Primarily C is the most connected knot to S, i.e. $\Psi(S,C)$ is maximal. Again, this choice is still, in general not uniquely determined. Let C1 and C2 be two equally connected elements to S. Then

$$\Psi(S,C1) = \Psi(S,C2),$$

let us also assume that $\delta(C1) < \delta(C2)$ and evaluate the degrees of the two possible coalescences

$$\delta(S \circ C1) = \delta(S) + \delta(C1) - 2\,\Psi(S,C1)$$

and

$$\delta(S \circ C2) = \delta(S) + \delta(C2) - 2\,\Psi(S,C2).$$

Therefore $\delta(S \circ C1) < \delta(S \circ C2)$ and since there is no more than one minimal coalescence to S, if it exists, it must be the one with the lowest resulting degree. And so C1 is chosen as the "closest" to S with lowest degree.

6.3. Algorithm

Since the secondary order is common for the selection of both S and C, it is sufficient to sort all knots on increasing degrees,so that S is the "first" of the smallest and C the "first" of the closest . Moreover, this sorting operation needs only be done once, over the set of all initial nodes. At each step, S and C are extracted from the order, and the resulting $S \circ C$ is inserted accordingly.

Fig. 13 is a structure diagram [Li77] of the tree building algorithm, whose input is the original graph and output the connectivity thus built.

```
Make nodes into knots
Sort on increasing degrees
    Select   Smallest
    Choose Closest
    Coalesce
    Reinsert in order
until only one
knot is left
```

FIG. 13 - Tree building algorithm

6.4. A practical example

To illustrate the method let us consider the network in figure 14. a) used in Luccio and Sami's work [LS69], p. 188, as an example of the minimal groups identification procedure.

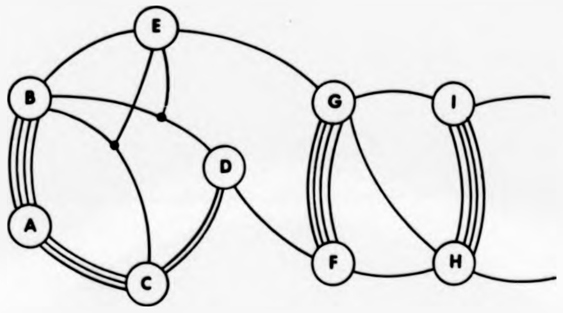


FIG. 14 a) Sample network

Converting to our net representation, where each edge in a signal set of size n is given the weight 2/n, and multiplying all weights by a factor of 3 for clarity, we obtain the graph depicted in figure 14. b).

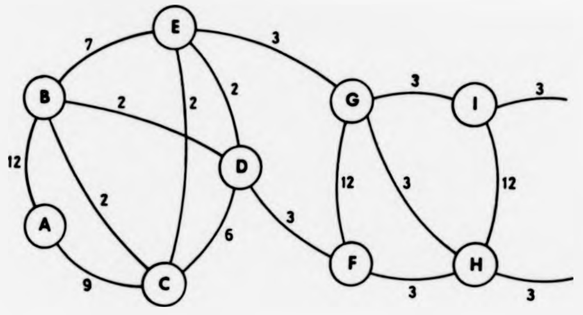


FIG. 14 b) - Sample graph to illustrate tree building algorithm

By evaluating the degrees of all nodes, and sorting on increasing order, we obtain:

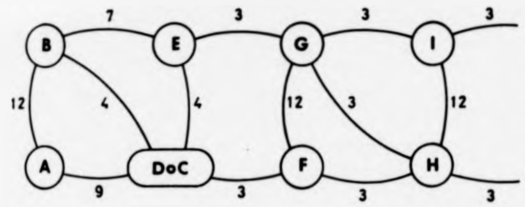
knots	D	E	F	I	C	A	G	H	B
degrees	13	14	18	18	19	21	21	21	23

The smallest element in the list is D,and C is the most connected to it. Coalescing and evaluating the resulting degree:

$$\delta(D \circ C) = \delta(D) + \delta(C) - 2\,\Psi(D,C) =$$

$$= 13 + 19 - 12 = 20\ .$$

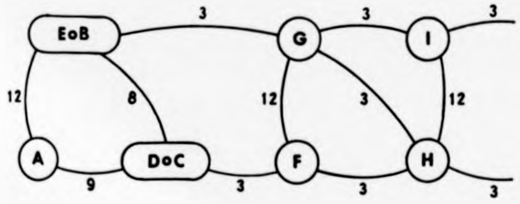
Figure 14. c) shows the graph after this first step, as well as the new order of knots



	★			D				
knots	E	F	I	C	A	G	H	B
degrees	14	18	18	20	21	21	21	23

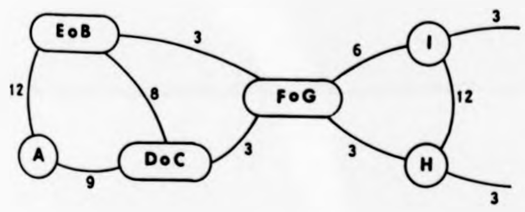
FIG. 14 c) After the first step

The process is then repeated until there is only one knot left, as depicted in figures 14 from d) to j).



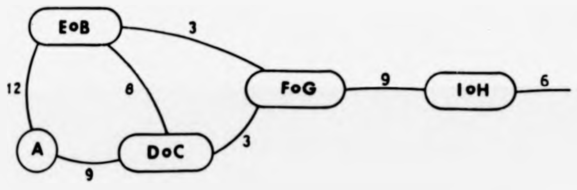
	★		D				E
knots	F	I	C	A	G	H	B
degrees	18	18	20	21	21	21	23

FIG. 14 d) After the second step



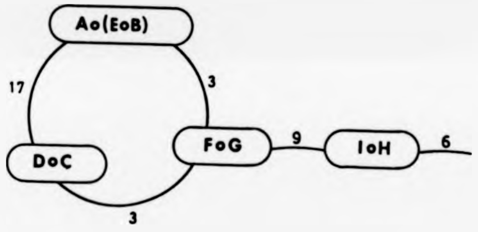
	F	★	D			E
knots	G	I	C	A	H	B
degrees	15	18	20	21	21	23

Fig. 14 e) After the third step



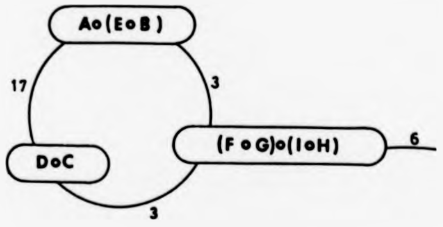
	F	I	D	★	E
knots	G	H	C	A	B
degrees	15	15	20	21	23

FIG. 14 f) After the fourth step



	★			A
	F	I	D	E
knots	G	H	C	B
degrees	15	15	20	20

FIG. 14 g) After the fifth step



	F		
	G	★	A
	I	D	E
knots	H	C	B
degrees	12	20	20

FIG. 14 h) After the sixth step

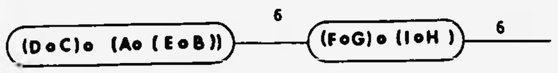


FIG. 14 i) After the seventh step

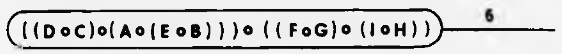


FIG. 14 j) After the eight step

After eight steps there is only one knot left, in this case with degree 6, since the given graph had two external connections of weight 3.

As the defined operation coalescence is not associative, the sequence in which those operations were performed determines uniquely the connectivity tree represented in figure 15. The figure on the left shows the degrees and the one on the right the ratios at each vertex.

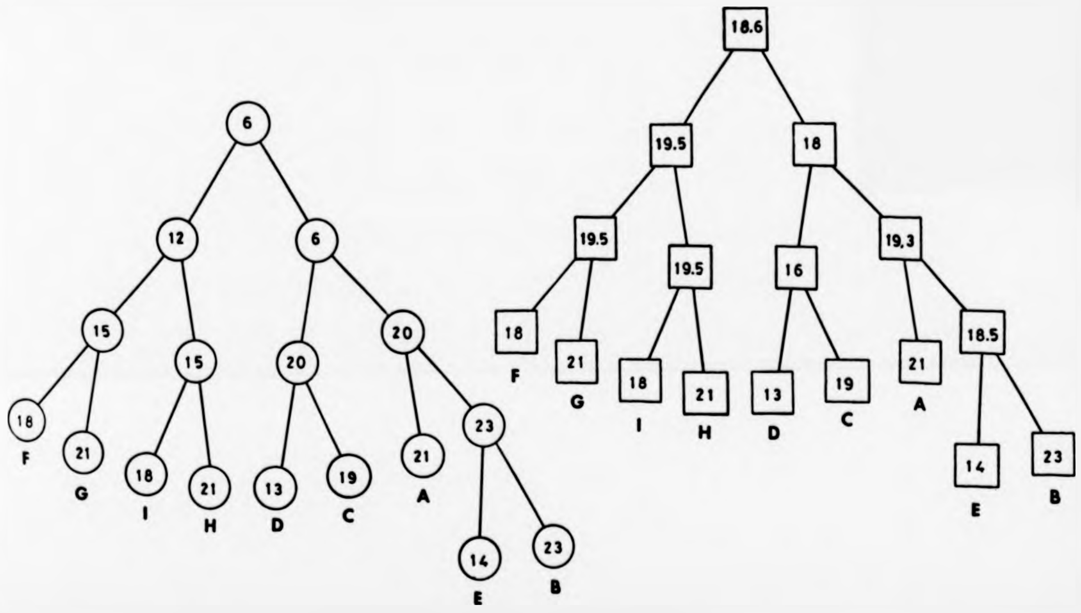


FIG. 15 - Connectivity tree for sample graph

On the whole, the three objectives are acceptably achieved. The tree is well balanced, degrees decrease upwards at most vertices and the uniformity of ratios is the best for this particular example. A better solution in terms of the min-dist objective, e.g. producing $D \circ (E \circ (C \circ (A \circ B)))$ instead of $(D \circ C) \circ (A \circ (E \circ B))$, would be unbalanced both in size and connectivity ratio. Luccio and Sami's procedure determines the minimal groups $\{A,B\}$,$\{F,G\}$,$\{I,H\}$ and also $\{\{A,B\}, C,D,E\}$ and $\{\{F,G\}, \{I,H\}\}$. This one would therefore be the best solution solely in terms of total minimal distance and when a binary structure is not required.

In general, the algorithm favours even distribution over minimal distance. A simple case is the one pointed out in figure 12 (§ 5.4) where the second solution will be generated, with $\rho(B) = \rho(A \circ C) = 4$.

As a final remark note that, in spite of the operation coalescence being commutative, the smallest knot S was always kept at the left hand side branch, so that at every tree vertex the left hand side branch is always smaller or equal in size to the right hand one.

6.5. Complexity

Due to its simplicity, the algorithm shows a limited growth in running time. For $\underline{n}$ initial nodes, a sorting process on increasing degrees is performed only once. Over the linked list of all elements in this defined order, three traverse operations are executed in each step: the search for S and C and the insertion of $S \circ C$. Note that the evaluation of $\delta(S \circ C)$ is independent of the total number of nodes. Since after each step the list becomes one element shorter at the end of n-1 steps the number of operations is

$$3n + 3(n-1) + \ldots + 3.2 = 3/2\ (n+2)\ (n-1)$$

i.e. a polynomial of degree 2. For that reason, a sorting method of order $\mathcal{O}(n^2)$ was considered adequate for the initial ordering on increasing degrees. Shakersort [Wi76], an improved version of Bubblesort alternating the direction of consecutive passes, was used in the implementation. We may therefore conclude that the described tree-building algorithm is of order $\mathcal{O}(n^2)$ in running time.

The algorithm also shows a limited growth in terms of memory space. Since a binary tree with $\underline{n}$ leaves has a total of 2n-1 vertices, the implementation of the tree structure only requires a number of (equal in size) blocks of order $\mathcal{O}(n)$.

The present phase of the placement method, as well as the following one, requires access to the connectivity matrix representing the graph model. This is a symmetric matrix of order $\underline{n}$ and often very sparse. Although not provided in the performed implementation, special storage techniques for sparse symmetric arrays might be considered for large problems.

CHAPTER IV

MAPPING THE CONNECTIVITY TREE

1. INTRODUCTION

At this stage of the proposed placement method, the hierarchical structure of a given circuit is represented in terms of a binary tree, which was built to pursue three main objectives: minimal total distance, even distribution of connections and symmetry at each vertex.

In the present Chapter we describe the mapping of the connectivity tree into the given board area. Two algorithms are presented, one for regularly structured boards with fixed locations for modules and the other for the general case of a continuous plane where components of dissimilar size may occupy any distinct positions. The problems raised by specific design requirements and by high component density are analysed in both types of environment.

2. BASIC PROCESS

In order to introduce the basic concepts involved in the mapping process, we make a number of assumptions defining an *ideal* board environment.

Assume a rectangular board of sufficiently large area, without fixed locations for modules, and also a set of components of similar size and shape. There are no design restrictions concerning the assignment of components to specific positions on the board.

2.1. Areas

Given a connectivity tree and an ideal board, the embedding of the tree on the board surface is a top-down hierarchical process which assigns tree vertices to rectangular board areas.

At a certain stage during the process, a tree vertex comprising a knot $A \circ B$ has already been assigned to a specific rectangle. Let us denote that rectangle by $[A \circ B]$ of which both the location and the dimensions are uniquely determined. $[A \circ B]$ is now going to be partitioned into two rectangles $[A]$ and $[B]$.

The first decision to be taken concerns the *direction* of the partition line. As a rule, the rectangle is partitioned across its longer dimension. The resulting rectangles will thus have minimum perimeter, therefore reducing the distances between components within each knot. Recall that the circuit elements belonging to knot A are expected to be placed closer to each other than to elements in B.

Next, the *areas* of the resulting rectangles [A] and [B] must be evaluated. Since components were assumed to be all similar in size, the area of each rectangle may be determined as proportional to the number of components, with

$$\text{area } [A] = \frac{\eta\,(A)}{\eta(A \circ B)} \text{ area } [A \circ B].$$

The area attributed to each knot is therefore proportional to the total number of components included, i.e.

$$\frac{\text{area } [A]}{\eta\,(A)} = \frac{\text{area } [B]}{\eta\,(A)} = \frac{\text{area } [A \circ B]}{\eta\,(A \circ B)} \;.$$

In this case, the board area occupied by components and consequently the empty spaces will be uniformly distributed on the board surface.

Since the connectivity tree contains at each vertex A the value of its total connectivity $\sigma(A)$, an estimate of the expected routing area may easily be included. The area assigned to the subrectangle A would then be determined as

$$\text{area } [A] = \frac{\sigma(A)}{\sigma(A \circ B)} \text{ area } [A \circ B]$$

so that

$$\frac{\text{area } [A]}{\sigma(A)} = \frac{\text{area } [B]}{\sigma\,(B)} = \frac{\text{area } [A \circ B]}{\sigma\,(A \circ B)} \;,$$

which represents a uniform distribution of connections at every area in the board.

Owing to the fact that a major objective in the building of the connectivity tree was an even distribution of connections, the latter evaluation of areas will not differ largely from the former one. Note that for a coalescence where $\rho(A) = \rho(B)$:

$$\frac{\sigma(A)}{\sigma(A \circ B)} = \frac{\rho(A)\ \eta(A)}{\rho(A)\ \eta(A) + \rho(B)\ \eta(B)} = \frac{\eta(A)}{\eta(A \circ B)} \ .$$

2.2. **Permutation**

At this stage the dimensions of the subrectangles [A] and [B] are defined but not their relative position. The most convenient *permutation* [A/B] or [B/A] must still be determined. By [A/B] we denote that A is placed to the left of (or above) B.

At the first level (tree root), assuming no additional constraints, the choice of permutation is irrelevant, but it becomes increasingly important as we travel down the tree, processing areas whose surroundings are already placed.

Both permutations are evaluated at each step and the most convenient one is adopted. The notation *value* [A/B] represents the penalty of placing A at the left of (or above) B. This value will necessarily be a function of the distance as well as the connectivity between components. Each knot is considered placed at the centre point of the corresponding rectangle and distances are measured from there. At each step, the point that locates $A \circ B$ will generate two new points for A and B according to the chosen permutation, as illustrated in figure 16.

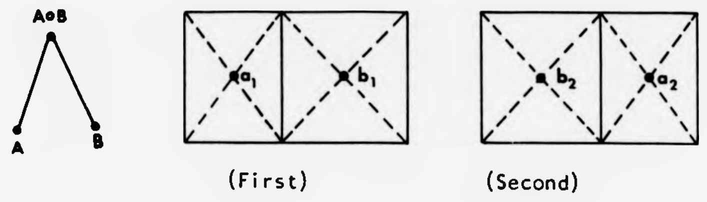


FIG. 16 - Choice of permutation

Let the first permutation be [A/B], where a_i and b_i are the corresponding centre points of [A] and [B]. The value [A/B] includes the *cost* of placing A at a_1 and B at b_1, i.e.

$$\text{value } [A/B] = \text{cost } (A,a_1) + \text{cost } (B,b_1).$$

Since all elements $x_i \in A$ are at a_1, the cost (A,a_1) will only depend on connectivity and distance to elements outside A. On the other hand, the relative distance from [A] to [B] is a constant for both permutations $(\overline{a_1 b_1} = \overline{a_2 b_2})$, as well as their interconnectivity $\psi(A,B)$. Hence, the terms in cost(A,a_1) representing connections within A and from A to B do not need to be evaluated.

By analogy with the concept of k-th order moment, as introduced in Chapter I. § 9, we define a *cost function* of order k as:

$$\text{cost}_k\ (A,a_i) = \sum_{x_i \in A} \ \sum_{x_j \notin A \circ B} c(x_i,x_j) .\ d^k(a_i,x_j)$$

where $d(a_l, x_j)$ is the distance between a_l and the centre point of the current rectangle containing x_j.

The adopted metric is a reduced rectilinear distance. As the partition line (and also the segment $\overline{ab}$) is always either vertical or horizontal, the component of the distance which is parallel to the borderline is a constant for both permutations. The evaluation of the distance may therefore be reduced to the component which is parallel to $\overline{ab}$. Note that, Euclidean distance and rectilinear distance are equivalent for the present purpose.

2.3. Squared distance

Based on the discussion in Chapter I § 9, the adopted cost function was the one in terms of a squared distance (k=2). In the present section we investigate the effect of such a function when compared to the corresponding one in terms of a linear distance (k=1).

Let us examine the simple example depicted in figure 17. a). The size of individual components is represented by units of the grid imposed on the board. Assume that $\eta(A) = \eta(D) = 2$ and $\eta(B) = \eta(C) = 6$. Knots C and D having already been placed, the present decision concerns the choice of permutation [A/B] or [B/A]. Figure 17 b) shows the resulting placement solutions.

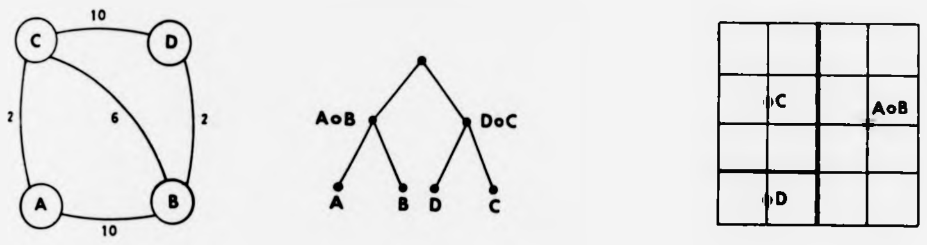
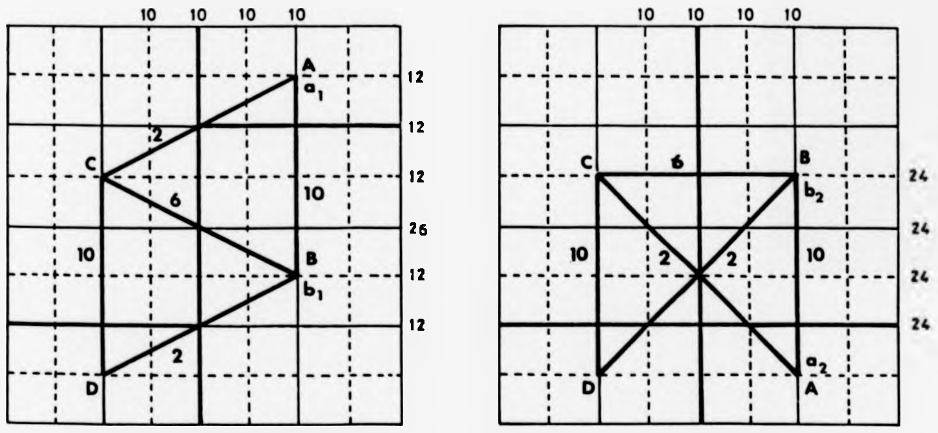


FIG. 17 - The effect of squared distance

a) Sample problem



b) Placement solutions showing crossover analysis

By evaluating the cost function based on a linear distance

$$\text{cost}_1\ (A,a_1) = \sum_{x_i \in A} \sum_{x_j \notin A \circ B} c(x_i,x_j)\ .\ d(a_1,x_j)$$

and taking the grid width as unit, we obtain for both permutations the values

$$\text{Value}_1\ [A/B] = \text{cost}_1\ (A,a_1) + \text{cost}_1\ (B,b_1)$$
$$= (2 \times 1) + (6 \times 1 + 2 \times 1) = 10$$

$$\text{Value}_1\ [B/A] = \text{cost}_1\ (A,a_2) + \text{cost}_1\ (B,b_2)$$
$$= (2 \times 2) + (6 \times 0 + 2 \times 2) = 8.$$

The second permutation would, in this case, be selected. On the other hand, by repeating the process with the cost function of second order, we determine

$$\text{value}_2\ [A/B] = \text{cost}_2\ (A,a_1) + \text{cost}_2\ (B,b_1)$$
$$= (2 \times 1^2) + (6 \times 1^2 + 2 \times 1^2) = 10.$$

$$\text{value}_2\ [B/A] = \text{cost}_2\ (A,a_2) + \text{cost}_2\ (B,b_2)$$
$$= (2 \times 2^2) + (6 \times 0^2 + 2 \times 2^2) = 16$$

which favours the first permutation.

Our method selects the first permutation and a human designer would probably do the same. In fact, the first permutation produces a better solution with respect to uniformity of distances as well as an even distribution of connections. The following table shows the values of the moments M_1 and M_2 as well as the variance of distances σ_d^2 (as defined in

Ch. I. § 9), for the total evaluation of both metrics:

	Euclidean			Rectilinear			
	M_1	M_2	σ_d^2	M_1	M_2	σ_d^2	M_0
[A/B]	62.36	130	0.012	70	170	0.222	30
[B/A]	63.31	136	0.079	68	168	0.462	30

The Euclidean distance, as including "squared" factors, presents lower values for the first permutation. As to the rectilinear metric, the total distance is larger but with a lower variance in the values of individual distances.

In order to perform a crossover analysis, a grid of cut lines with a width equal to half the unit of distance was imposed on the board, as represented by dashed lines in figure 17.b). Crossovers are counted in such a way that each connection with endpoints over two lines, only counts at one of them. Hence, there is a total of 16 cut lines. By evaluating the weighted crossover moments X_1 and X_2 as well as the corresponding variance, for both permutations, we obtain the following values:

	X_1	X_2	σ_x^2
[A/B]	140	2328	68.937
[B/A]	136	2704	96.750

As pointed out in Chapter I. § 9, a simple count of crossovers (X_1) is identical to a rectilinear total distance. However, to minimize X_2 ensures a uniform distribution of these crossovers on the set of all cut lines.

The proposed formulation of the cost function does therefore ensure an adequate choice between both possible permutations, while only requiring a limited amount of computations.

Note also that the "squared" distance has the effect of reinforcing the growth of the cost function with an increase of distance. The above example was selected for its predominance of connectivity (weights) over distances, which caused the observed disparity of some results. A parallel study for a similar example with all weights equal to one would show the first permutation as the one producing lower values in all the tests performed.

2.4. The method

At this stage both rectangles [A] and [B] are uniquely determined by their dimensions and location. If neither comprises isolated components the process is recursively applied to both of them. The decision on which rectangle is going to be processed first defines the order of the connectivity tree traversal during the mapping phase.

As a rule, priority is given to the rectangle with the larger number of components. In this way the bigger rectangles are partitioned first, as the smaller the area the more accurate is the location of its components. Moreover, since a major objective in the building of the connectivity tree was a constancy of the ratio $\rho(A) = \sigma(A)/\eta(A)$ at each vertex A, the bigger rectangle will also be the one comprising the knot with higher connectivity. The first single component to be placed on the board is therefore the one of higher degree. The process is then repeated in a top-down depth-first order until each rectangle contains one single component.

Figure 18 summarizes the main steps in the basic form of the tree--mapping process.

```
MAP(tree vertex; rectangle)
    Choose direction
    Evaluate areas
    Select permutation
    Choose next vertex
    Recursive calls
```

FIG. 18 - Basic tree-mapping process

The described method does not consider design requests for the specific location of certain components. Yet, the particular case of preplaced external connectors may be easily implemented. They are assigned to fixed coordinates at the edges of the initial rectangle, from which the distances to connected components are measured.

In its basic form the tree-mapping method is limited to the type of placement problems where all modules are of similar size and the board is a continuous surface without fixed places for components. In the remainder of the present Chapter we analyse the utilization of the method in two special types of environment.

3. REGULARLY STRUCTURED BOARDS

In this paragraph we study the case of regularly structured boards with fixed *slots* for modules, separated by routing *channels*.

Let $\underline{m}$ be the number of slots and $\underline{n}$ the total number of components, with $n \leq m$. It is assumed that each component may be assigned arbitrarily to any slot and also that the slots are disposed along equidistant rows and columns. A grid may thus be imposed on the board, in such a way that each rectangle (not necessarily a square) comprises one single slot. The location of each slot is defined by the coordinates of the centre point of the corresponding rectangle.

Strictly for the evaluation of rectangular areas, a coordinate system is employed where the unit of length on the x-coordinate (y-coordinate) is the width (height) of the mesh. At a given stage during the process, a rectangle $[A \circ B]$ is to be partitioned into two rectangles $[A]$ and $[B]$, in such a way that both dimensions of all rectangles have integer values. Let $S[A \circ B]$ be the longer dimension of the given rectangle, which is to be divided into two portions $s[A]$ and $s[B]$. These values are made proportional to $\eta(A)$ and $\eta(B)$ respectively and rounded to the next integer, as

$$s[A] = \text{round}\left(\frac{\eta(A)}{\eta(A \circ B)}\, S[A \circ B]\right)$$

$$s[B] = S[A \circ B] - s[A]$$

Clearly, the desirable proportionality of areas is not exact, especially for small values. This fact tends to cause conflict situations, in particular at the lower hierarchical levels in very dense boards.

3.1. Space conflict

Consider the situation depicted in figure 19, where $\eta(A) = \eta(B) = 3$. The evaluation of lengths gives

$$s[A] = \text{round}\ (\tfrac{3}{6} . 3) = 2 \quad \text{and} \quad s[B] = 1,$$

which makes [B] insufficient for all components in knot B.

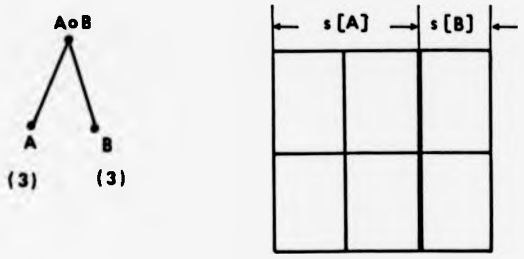


FIG. 19 - Space conflict example

When such a conflict is detected, the algorithm will modify the connectivity tree in order to make it fit the board geometry. The remaining space in [A], denoted free [A], is estimated and a search is made in B for an element to be transferred to A. Note that at the previous level of recursion the whole [A o B] was ensured to be sufficient for all elements in A o B.

Starting at vertex B a tree traversal is made towards the lower levels, searching for an element E able to fit free [A]. The first E such that $\eta(E) \leq$ free [A] will be transferred to A, even if $\eta(E)$ is bigger than the necessary to make B fit into [B]. The loop is necessarily bounded since the excess in [B] is never greater than the remaining space in [A].

Recall that, in spite of the operation coalescence being commutative, the connectivity tree was built in such a way that, at each vertex, the left hand side branch is always smaller than or equal to the right hand one (Smaller joins Closest). Consequently, if the search for E starts always at the *left* hand side branch, the process will be faster and also less likely to find the most heavily interconnected elements. Detailed experimentation confirmed this assumption, showing that a preorder search for E along the left branches first, is quicker and produces better placement results.

Once the element E is found, it is deleted from its place in the tree and "subtracted" at each vertex on its way up towards B. Then, it is appended to the branch starting at vertex A and accordingly "added" to the intermediary vertices. The transferred element is appended to a terminal vertex of the branch starting at A, giving priority at each vertex to the *smaller* branch. This makes the process faster and also less likely to disturb the heavily interconnected branches. Experimentation showed that a search for the element in A which is more connected to E, is much slower and does not improve the final placement results. Note that, since the cost function grows very rapidly with distance and E is more strongly connected to B than to A, it will in most cases be placed in the neighbourhood of [B].

At the end of the smallest branch, the transferred element is appended at the left hand side, so that it is the first to be chosen if a further alteration is needed at a lower level. This minimizes the number of elements wich are disturbed from their position in the original connectivity tree.

3.2. Proportionality

In the last paragraph we studied the process of fitting the connectivity tree to the particular board geometry, in order to solve space conflict situations. The usage of the process may however be generalized to other problems. Suppose that a similar sub-tree to the one used in the former example is to be mapped into the board area represented in figure 20.

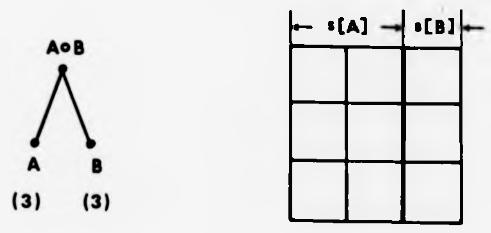


FIG. **20** - Loss of proportionality

Both rectangles are now sufficient, but the proportionality of areas is lost. A check of proportionality may be introduced in order to detect this type of situations and accordingly modify the tree, not to solve space conflicts, but to keep areas proportional to the number of components.

This procedure will anticipate congestions and also space conflicts at the forthcoming levels.

The statement made in Chapter III § 5.3, on symmetry being a main objective in the building of the connectivity tree, is now clear. A tree vertex where a knot of very small size was coalesced to a much larger one, would necessarily produce a long sparse area at the side of a congested one.

3.3. Preplacement

The next question to be analysed concerns the existence of components whose location is totally or partially predetermined. The tree mapping algorithm recognizes two main types of elements in each knot A. The *floating* components (float(A)) for which the location is defined by the system and the *preplaced* ones (prep (A)) with given precise coordinates.

The decision on which is the most convenient permutation has, so far, been taken on connectivity only. However, since the problem is essentially the partition of a rectangle into two, it is possible to choose the knot containing a preplaced element to occupy the area including the predefined location, as will be discussed later.

The tree traversal order is also modified: branches with preplaced components are placed first. They will act as fixed points on the choice of permutation. The cost function cost (A,a_l) must also be adapted accordingly:

$$\text{cost}\ (A,a_i) = \sum_{x_i \in \text{float}(A)} \ \sum_{x_j \notin \text{float}(A \circ B)} c(x_i,x_j)\ .\ d^2(a_i,x_j).$$

this expression measures the weighted square distance from the elements in float(A) (considered placed at a_I) to all the others, including the preplaced ones in A. The whole knot A will thus tend to occupy the area where the prep(A) are located.

Once the permutation is selected, a verification is made for every preplaced element in both A and B. If any of them happens to be on the wrong side, it is deleted and transferred to the other branch. This will not alter the permutation, since it only depends on the floating components.

This alteration of the tree represents the fact that the predefined placement of components may be incompatible with connectivity. Design requirements are then satisfied to the detriment of the structure defined by the connectivity tree. Yet, if a particular group is strongly connected to a preplaced element, it will be kept in its neighbourhood since the cost function grows very rapidly with distance.

A very frequent example of preplacement is a row of external connectors at one (or more) of the board edges. *Edge connectors* are treated as "semi-placed" components: one coordinate is fixed (e.g. $y = 0$) and the other may take any value along the edge. The obvious solution is to define them as preplaced at the steps where the boundary line is parallel to the edge, and as floating when that line is perpendicular to the edge.

3.4. Algorithm

In figure 21 we summarize the main steps of the tree-mapping algorithm for regularly structured boards with fixed slots for components.

```
MAP (tree vertex; rectangle)
    Choose direction
    Evaluate areas
    Select permutation
    Check preplacement
    Solve space conflict
    Adjust proportionality
    Choose next vertex
    Recursive calls
```

FIG. **21** - Tree-mapping process for regularly structured boards

The evaluation of the area attributed to each subrectangle is also adjusted in order to take preplacement into account. Preplaced elements, although incorporated in the tree, are not included in the counting of components at each tree vertex. The actual positions they occupy on the board are included in a list of all the forbidden slots as required by the designer. Sizes of rectangles, in both permutations, are therefore evaluated in terms of the available slots for floating components. Adjustments of the connectivity tree in order to solve space conflicts or to balance the number of components in both rectangles must also be performed accordingly.

Since in the majority of the problems the whole set of modules does not cover every available slot, at the end of the process some components will have been assigned to two or more possible locations. In order to select the most favourable slot for each one of those components, a final step must still evaluate the cost function at every possible location.

4. GENERAL CASE

In the present paragraph we discuss the tree-mapping process in its general form. The board is a continuous rectangular plane of limited area, where components of dissimilar size and shape occupy any distinct positions, subject to a number of design constraints.

4.1. Evaluation of areas

When compared to the basic process, the area allocated to a particular knot A is no longer a direct function of $\eta(A)$, since it must account not only for different sizes of components but also for the necessary routing area. Prediction of wiring space depends on factors such as track-spacing which forms part of the design rules of individual technologies. An approximation of the area attributed to each knot A may be evaluated as a function of the total area occupied by components and the connectivity $\sigma(A)$, adjusted by a factor depending on the particular technology in use. A more accurate evaluation could be associated with the tree building process, by estimating the width of the interconnection channel between two coalescing knots.

At the tree root we have an approximate measure of the minimum total area required by the complete layout of the circuit. This information can be made available to the user at this stage, allowing for the detection of impossible cases before an attempt at placement is actually performed. By making full use of this property, the connectivity tree could also become a valuable tool in the partitioning of a circuit into different boards.

At a given stage during the process, let $\mu(A)$ and $\mu(B)$ be the *minimum total area* estimated for the layout of knots A and B respectively. The direction of partition follows the basic rule and subrectangles sizes are determined in function of the given area [A o B] in such a way that

$$\frac{\text{area }[A]}{\mu\,(A)} = \frac{\text{area }[B]}{\mu\,(B)} = \frac{\text{area }[A \circ B]}{\mu(A) + \mu(B)} \;.$$

It should however be pointed out that, the minimum area estimated for a given knot may only be achieved by means of a packing process. In a binary partition mode, the dissimilarity in components sizes as well as the presence of preplaced elements leads to a certain degree of wastage, therefore requiring a larger area than the estimated minimum. As the next two sections describe, the implemented version of the method is compatible with the most frequent cases of preplacement and, to some extent, adjusts the disposition of components is order to fit a tight area.

4.2. Preplacement

The estimated minimum total area $\mu(A)$ of a given knot A comprises the area occupied by the floating as well as the preplaced components in A. External connectors are not included; they will be placed on a strip along the edge(s) outside the rectangle representing the effective board area.

The order in which the connectivity tree is traversed and also the selection of the most favourable permutation are adjusted in terms of the preplaced elements, in a similar manner to the one used on regularly structured boards.

Once the permutation is selected, a test is made for the location of the centre point of every preplaced component. Those elements which happen to be on the wrong side are transferred to the other branch as in the preceding case. The whole process is checked from the beginning by re-evaluating the subrectangles areas, while keeping the same permutation.

Since only the centre points of preplaced components were guaranteed to be on the correct side, a subsequent step must still check the boundary line position in relation to the area occupied by each preplaced component. If the borderline intersects any of those elements, it is moved accordingly. Both subrectangles are checked and, if no acceptable position is found for the boundary line, a second attempt is performed by changing the *direction* of partition.

This simple approach proved adequate for the most frequent practical problems. In exceptional cases, where it is impossible to draw a partition line without crossing preplaced components, the problem may be solved by replacing groups of fixed elements by whole preplaced blocks.

4.3. Space fitting

As already pointed out, the implemented tree-mapping method does not execute the packing of components into as small an area as possible. However, it performs a certain degree of adjustment between the shape of the connectivity tree and the available board area. Such a process is based on the following assumption: "A given area is sufficient for the placement of a knot A, if it accommodates a regular arrangement of components of average size as well as the largest module in A".

This procedure is solely intended to adjust the disposition of components, as defined by the connectivity tree, towards the shape of the available board area. In the case of a very dense assembly, the method requires the placement of the circuit on a larger virtual area, followed by a compaction routine.

Although the area attributed to a given knot A is larger than the total area of its components, it must still be ensured that such space is sufficient to accommodate a particularly long module. The *longest* module in A is defined as the one with maximal dimension, in the direction which is perpendicular to the partition line. Note that the other dimension has already been verified at an earlier level.

The present test is associated with the one that checks the borderline position in relation to the preplaced elements. Whenever the line intersects the longest component, it is moved accordingly. Similarly, if the opposite subrectangle is also insufficient for the longest module in B, a second attempt changes the boundary line direction.

Once the borderline position is acceptable, both in terms of the longest modules and the preplaced elements, the resulting subrectangles areas are tested for a regular arrangement of average components. Both dimensions of the *average* component in A are evaluated as the arithmetic mean of the corresponding dimension for all components in knot A.

This test comprises a collection of heuristics, intended to fill empty areas generated by the previous borderline displacement and also to adjust the relative positions of components as an attempt to fit into a particularly densely occupied space.

As an example of the first problem, let us consider the simple situation depicted in figure 22 a). Knot A comprises the larger module and knot B the three smaller ones. The borderline displacement created an insufficient area for all components in B. A grid of average components from B, imposed on the empty area in [A], allows for the approximate evaluation of how many modules need to be transferred. This process is similar to the one described for regularly structured boards, but with the important difference that the actual sizes of transferred elements must be verified.

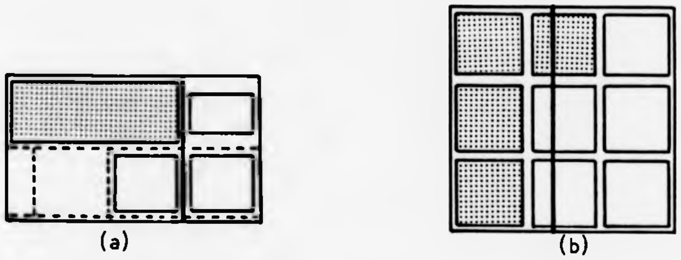


FIG. **22** - Space fitting examples

The second type of space conflict is caused neither by preplacement nor by a dissimilarity in component sizes. However, it tends to occur, particularly at the lower stages of the process in very dense boards. As an example, assume that nine modules of similar size are to be placed on a square area, with $\eta(A) = 4$ and $\eta(B) = 5$, as shown in figure 22.b). The borderline position, according to the general rule, generates insufficient areas at both sides. Note, that a grid of average components in A o B has been tested in [A o B] at a former level.

At the present level, a grid of average A-components shows that [A] is insufficient: 3 effective spaces for 4 modules. Likewise in [B], there are 3 spaces for 5 modules. When this situation is detected, the space fitting procedure searches for an element in A, which is able to fit the grid in B. The element is transferred and rectangles areas are reevaluated. As a rule, the element is transferred from the smaller to the larger knot (excepting when the smaller comprises a single component) since the conflict would otherwise be repeated at the next step, therefore generating an infinite loop.

The use of the space fitting procedure may also be generalized in order to ensure the desirable proportionality of areas.As an example, the presence of an unusually large module may generate a sparse board area, even if no conflict is detected in the opposite rectangle. This situation must be detected as early as possible and consequently rectified, not only to achieve proportionality but also to avoid forthcoming space conflicts.

Yet, it is debatable whether this should be performed as a systematic process or only when the imbalance is considerable, since the connectivity tree should ideally be kept unaltered. We suggest that the systematic process is introduced only in a second attempt on a critical board. This approach was tested on a particularly dense board and proved successful.

4.4. Algorithm

Figure 23 summarizes the main stages in the tree-mapping algorithm in its general form

```
MAP (tree vertex; rectangle)
    Choose direction
    Evaluate areas
    Select permutation
    Check preplacement
    Check boundary line
    Space fitting
    Adjust proportionality
    Choose next vertex
    Recursive calls
```

FIG. **23** - General case of tree-mapping process

The growth of the process in terms of running time depends heavily on the number of tree alterations performed. The basic binary tree traversal ranges from $O(n \log_2 n)$ when perfectly balanced, to $O(n^2)$ in the worst case. At a given tree vertex with k components, where no space conflict arises and no preplaced element is transferred, the process is linear, i.e. $O(k)$ at each step. Note that, in this case, only the evaluation of areas and the cost function depend on k. When tree alterations are required, the growth in running time depends on the number of transferred elements, rather than the total number of components k. Detailed experimentation showed that, when only a few elements are transferred, the process is kept near-linear at each step, as will be shown in the next Chapter.

CHAPTER V

IMPLEMENTATION AND RESULTS

1. INTRODUCTION

In this Chapter we report a summary of the results obtained on a practical implementation of the proposed placement method. The main sections correspond to the two basic types of environment: regularly structured boards and the general case.

The numerical results comprise observed values of objective functions, running times and rates of successfully routed connections. Different versions of the basic method, as well as manual placement solutions, are also assessed and discussed.

2. IMPLEMENTATION

The proposed placement method, as described in the last two Chapters, has been implemented in Pascal on the CDC 7600 of the University of Manchester Regional Computing Centre. A simple interface with the existing CAD system has enabled direct access to data files holding the circuit description of real boards, which happened to be being designed at the time. These are wire wrap and PC boards, part of the hardware of mainframe MU6G, whose implementation layout was executed by members of staff and research students.

The following report includes, as a reference, the corresponding results obtained by manual placement. For the magnitude of the studied problems, it seems adequate to establish a comparison between automatic and hand placement. With one or two hundred components, given the time and expertise, the human designer will always be able to produce better results. The aim of layout automation is primarily to reduce the design time as well as the number of errors. An automatic placement system should therefore achieve, in the shortest possible time, results which are comparable with those obtained by manual placement.

True evaluation of placement is determined by how eficiently it can be routed. However, the routability of an automatic placement solution is at least as dependent on the tracking scheme used as a manual placement. The available routing system was reasonable but, at the time, still under development.

The router used [Lo84] was designed as part of the Generalized Tracking System, developed at the Department of Computer Science at the University of

Manchester. It consists of an improved version of Aramaki's method [AK71] and can be briefly stated as follows: when shearching for a route between two pins, draw a horizontal line from each pin and extend it as far as possible. An unblocked vertical line is then looked for, connecting the two horizontals. The process can also be performed along the opposite direction.

This tracking system was also sensitive to net ordering and a systematic study of the most effective sequence of nets, for each board, was beyond the scope of this thesis. A number of placement solutions was sent to the tracking system, together with those obtained manually, and some of the results will be reported in the following paragraphs.

The main discussion in the present Chapter considers aspects which are measurable directly from the obtained placement solutions. These comprise objective functions as defined in Chapter I.§9, as well as the observed running times in relation to the estimated complexity of the algorithms involved. Comparative results on different versions of the basic method, which led to the taking of decisions on some points discussed in the earlier text, are also reported here.

3. REGULARLY STRUCTURED BOARDS

The most easily available information on the type of regularly structured boards with fixed slots for components, concerned the circuit description of a number of wire wrap boards known as Augat boards.

Augat boards accomodate a combination of 16 pin and 24 pin ECL logic. Each board comprises an array of 6 x 30 slots for 16 pin modules, with 6 x 2 locations for 24 pin Edge CONnectors (ECONs) along the top edge, as shown in figure 28. Particular locations in the basic array can also be used for 16 pin eXtra CONnectors (XCONs).

Since ECL logic requires terminal network resistors, locations for plugable single-in-line resistor modules are also available. As pointed out in Chapter I.§ 8, the chosen representation for nets in terms of complete graphs allows for net decomposition to be performed as a result of the obtained placement solution. The given sequence of components in each net may thus be rearranged and the terminal module identified, in such a way that the total net distance is minimized. A location for a resistor in then selected in the neighbourhood of the terminal module.

Augat boards have internal power layers, so neither placement nor the tracking system need to be concerned with space for track power connections on the board.

For a given board, the data file consists of a net-list in terms of pin-to-pin connections. The initial phase of the method, converts this information into a *graph model* representing the circuit, as discussed in Chapter I. § 8. Next, the *connectivity tree* is built as described in Chapter III.

3.1. Tree adjustments

In the present section we report a set of results obtained on the placement of two Augat boards, which refer to the tree mapping and subsequent tree adjustments discussed in Ch. IV . § 3. The following table specifies the magnitude of Augat boards AB1 and AB18: number of 16 pin components, external modules and nets.

	# components	# ECONs	# modules	# nets
AB1	109	7	116	579
AB18	99	3	102	302

The observed running time (in seconds), for the connectivity tree building was, for both boards:

	# modules	time
AB1	116	7.425
AB18	102	5.681

These values are in line with the estimated complexity of $\mathcal{O}(n^2)$ for the tree building algorithm. We may therefore establish an approximate evaluation of the required time to build a connectivity tree with n initial nodes, as

$$T_{tree}(n) = \alpha . n^2 \quad \text{with} \quad \alpha \approx 0.0005 \text{ sec}$$

which was supported by similar observations on other boards.

The basic connectivity tree may subsequently be adjusted in order to fit the available board area. As discussed in the last Chapter, the running time for the tree mapping phase will depend not only on the number of modules, but also on the amount of necessary tree alteration . In order to investigate the effect of such alterations on the total running time, we forced the given circuits into smaller sections of the basic Augat board. The following values are total job times, comprising the net-list input, graph model, tree building, tree mapping and output of the placement solution.

Board area	6 ×30	6 ×25	6 ×20	6 × 19
AB1	10.973	10.921	11.146	11.151
AB18	8.922	8.872	8.867	8.866

These results show that the increase in running time motivated by tree alterations is very limited. Such increase may even be overcompensated by the final step which selects the most favourable location for components whose domain comprises more than one slot. This fact explains the apparent disparity on the results of AB18, since a sparse board produces larger domains for its modules. On the other hand it becomes clear that, for this type of board, the major factor in the total running time is the building of the connectivity tree.

The former values were obtained with the tree adjustments as follows (Ch. IV . § 3.1): each transferred element is "searched" along the *smaller* branches of the origin knot, and likewise "appended" to a terminal vertex along the *smaller* branches of the destination knot. The comparative efficiency of this approach was supported by detailed experimentation, from which we select the following set of results obtained on the same boards AB1 and AB18.

The "search" along smaller branches was compared to the opposite solution, i.e., giving priority to the *larger* branch at each vertex. The effect of "appending" the transferred element to its *most connected* component at the destination knot, was also investigated. The four possible combinations were tested and evaluated.

Figure **24** shows the results on the placement of AB1 and A18 on the entire 6×30 Augat board area, as well as those obtained on the manual solution. These values comprise the total weighted Euclidean distance (dist), measured from the centre point of components, taking the grid size as unit, and the maximum single length (max). A weighted crossover analysis was also performed, using the same grid: $\bar{x}_v$ represents the average number of crossovers on vertical cut lines (likewise $\bar{x}_h$ on horizontal lines) and X_2 the total sum of squared crossovers. The total job times (in seconds) include a search for the nearest available location for a *resistor* at the terminal module of each net. The sequence of components in each net was also rearranged as a result of the layout solution. Starting from the initial module, the nearest one was searched within the net elements, and the process repeated until the terminal module was reached. The nearest available resistor for this module was then selected and added to the output of the new net sequence.

The set of results reported in figure 24 shows that, for average problems, the type of tree adjustments performed does not affect the final solution in a noticeable manner. The only observed differences are originated by the "search" procedure used, but these are not sufficiently conclusive. For that reason, the whole process was tested on a smaller section (6×20) of the board and the corresponding results were as shown in figure 25.

From this new set of values, it becomes clear that the most favourable strategy for tree adjustments is to give priority to *smaller* branches in

Board AB1 (6 x 30)

"search"	"append"	time	dist	max	$\bar{x}_v$	$\bar{x}_h$	X_2
smaller	*smaller*	11.826	3002.29	26.01	82.65	148.71	469054
smaller	*connect.*	11.824	3002.29	26.01	82.65	148.71	469054
larger	*smaller*	11.880	3044.55	26.01	84.44	148.42	472724
larger	*connect.*	11.871	3044.55	26.01	84.44	148.42	472724
(Manual Placement)			2993.64	19.10	72.58	150.00	380401

Board AB18 (6 x 30)

"search"	"append"	time	dist	max	$\bar{x}_v$	$\bar{x}_h$	X_2
smaller	*smaller*	9.884	1464.30	19.00	40.17	78.85	131663
smaller	*connect.*	9.867	1464.30	19.00	40.17	78.85	131663
larger	*smaller*	9.905	1464.08	19.41	39.89	79.14	130669
larger	*connect.*	9.889	1464.08	19.41	39.89	79.14	130669
(Manual Placement)			1977.08	23.02	56.72	60.28	181907

FIG. 24 - Effect of different tree adjustments on the entire board

Board AB1 (6 x 20)

"search"	"append"	time	dist	max	$\bar{x}_v$	$\bar{x}_h$	X_2
smaller	*smaller*	12.028	2350.02	16.03	84.36	168.00	401581
smaller	*connect.*	12.868	2547.59	16.03	94.63	168.42	462513
larger	*smaller*	12.189	2509.33	19.41	93.43	162.85	423144
larger	*connect.*	13.670	2469.23	17.11	91.63	167.57	428380

Board AB18 (6 x 20)

"search"	"append"	time	dist	max	$\bar{x}_v$	$\bar{x}_h$	X_2
smaller	*smaller*	9.825	1089.24	12.16	40.75	75.42	101350
smaller	*connect.*	10.265	1190.28	13.60	45.68	74.85	108354
larger	*smaller*	9.905	1180.86	13.03	44.15	82.71	118220
larger	*connect.*	10.619	1208.32	13.34	46.57	77.14	117487

FIG. **25** - Different tree adjustments on a smaller board area both operations. Besides being faster, it also produces better placement solutions. With respect to the "append" operation, as pointed out in Ch. IV § 3.1, a search for the most connected element at the destination knot is much slower and does not improve the final results.

Due to the way in which the connectivity tree was built, the smaller branch at each tree vertex is also the left hand side one. This property suggested a new form for the implementation of the "search" procedure: to give priority at each tree vertex to the *left* hand side lower branch. Moreover, since transferred elements are appended also to the left of terminal branches, this approach would minimize the number of components which are disturbed from their positions in the original tree. In fact, this alteration improved the observed placement results, although with a slight increase in running time. The comparative results of both approaches are reported in the following section.

3.2 Preplacement

The present section concerns the placement of Augat board AB6. This circuit has a total of 166 components, including 12 edge connectors (ECONs) and 40 external modules (XCONs), and a total of 861 nets. XCONs are 16 pin modules which are to be placed in the basic 6 x 30 array, near the board edges, in such a way that normal components do not occupy locations between XCONs and the nearest board edge. This board was selected in order to test the preplacement capabilities of the mapping algorithm. With basis on the manual placement solution, the 40 XCONs were treated as preplaced components and 9 locations which were left unoccupied for the above reason, were defined as forbidden areas.

In the placement of AB6, other aspects of the method were also investigated. The first subject concerns the used *graph representation* of the given circuit. As described in Chapter I. § 8, each net is represented by a complete graph connecting all its n elements with the weight w = 2/n attributed to single edges. In order to ascertain the benefit of such representation when compared with the basic model, where all edges are given the weight w = 1, both solutions were tested and assessed.

With respect to the obtained connectivity tree, for most of the observed practical problems, the graph model with w = 2/n generates a more unbalanced tree. This fact is due to two main reasons: firstly because the tree building algorithm tends to coalesce groups of equally interconnected elements into "long", i.e. unbalanced branches. Secondly, this effect is emphasized by the attribution of weight w = 2/n to single edges, which is intended to produce a more uniform distribution of weights on the whole graph. However, since the relative positions of components within an equally interconnected block are irrelevant for this type of boards, the only practical consequence is a slight increase in running time at the tree mapping phase. In fact, this representation produced better placement solutions in all the regularly structured boards tested.

Figure **26** shows the connectivity tree for AB6 obtained on the graph model with w = 2/n. Edge connectors (ECONs) are indicated by ○ and external modules (XCONs) by ● .

Another detail which was investigated in the placement of AB6 concerns the adopted "cutting" direction for *square* areas. Rectangles are as a rule partitioned across the longer dimension, but the chosen direction for

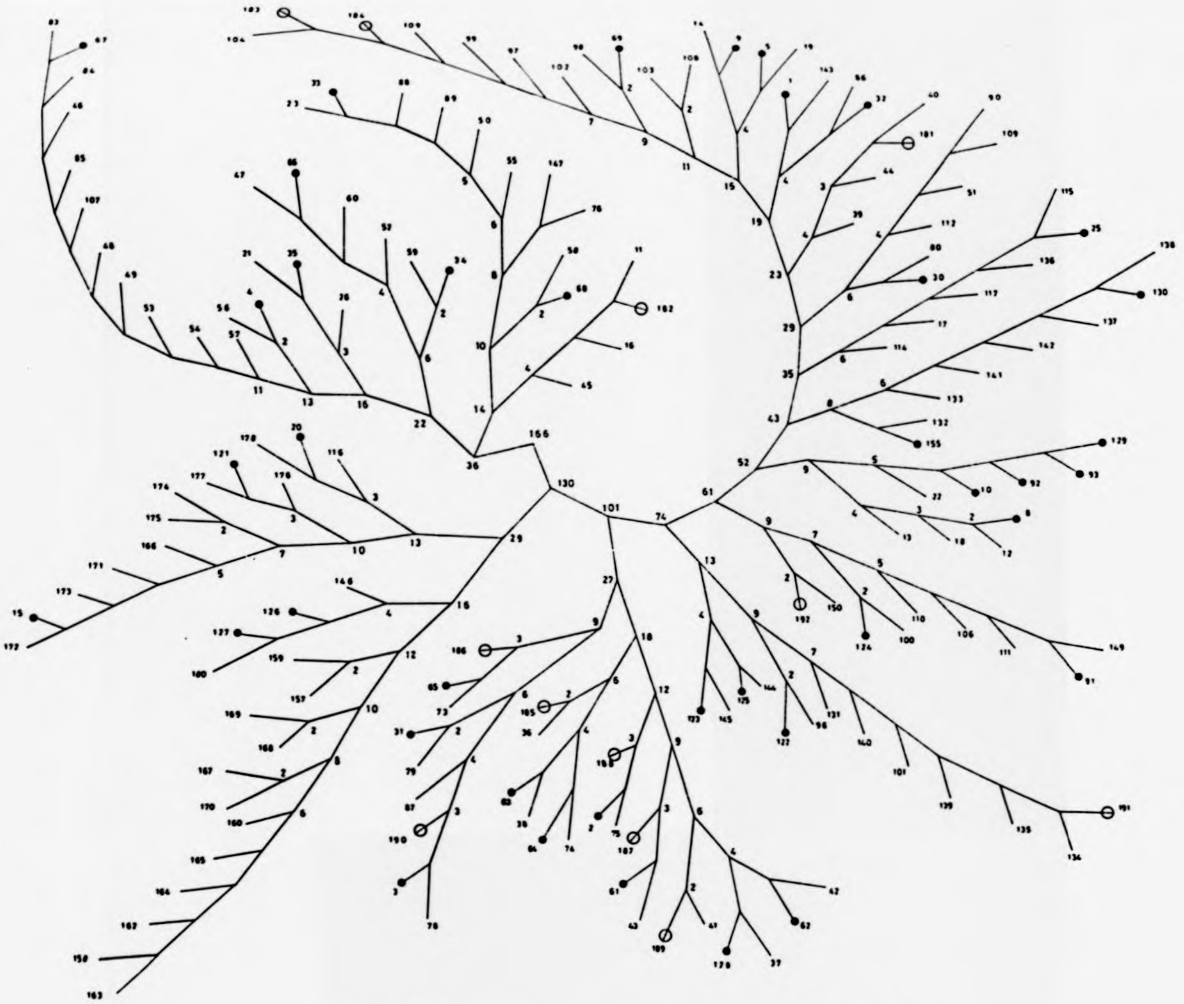


FIG. 26 - Connectivity tree for AB6 with w = 2/n

squares was proved to affect the placement solution. Recall that, in the coordinate system used, a *square* does not necessarily have equal dimensions in Euclidean metric. For that reason and also because of the Augat board geometry, the placement solutions were noticeably different for both cases. In general, the best results were obtained by "cutting" the square areas horizontally, i.e. across the longest grid dimension.

Figure 27 comprises two sets of results on the placement of board AB6. In the first set, only the 12 edge connectors were defined as preplaced components. A comparison between both graph models shows that $w = 2/n$ produces lower values with respect to total distance and number of crossovers. These solutions were obtained with the type of tree alterations suggested at the end of the last section: a "search" along the *left* hand side branches. The corresponding placement solution obtained on the previous "search" strategy was also tested for the $w = 2/n$ model. Although obtained in a lower running time, that solution did not improve the values of the basic objective functions.

The second set of results corresponds to the placement of AB6 with the 40 external modules as well as the 12 edge connectors defined as preplaced components, and with 9 predefined locations kept unoccupied. With respect to the used graph model, the lower values of distance and crossovers were obtained with the weights distribution $w = 2/n$. As to the cutting direction of square areas, the *horizontal* solution was observed to produce better placement results. Also, a comparison with the previous "search" strategy showed that those results were not improved.

On the whole, the placement solutions obtained by the proposed method

Board AB6

(Preplaced 12 ECONs)

	time	dist	max	$\bar{x}_v$	$\bar{x}_h$	X_2
w = 1	24.842	3765.43	21.00	105.31	215.85	722517
w = 2/n	25.603	3709.03	26.01	102.31	206.42	670220
" ("search" *smaller*)	21.690	3711.97	25.31	101.41	210.00	686867

(Preplaced 12 ECONs + 40 XCONs + 9 forb.slots)

"squares cut"	time	dist	max	$\bar{x}_v$	$\bar{x}_h$	X_2
w = 1 *vert.*	22.976	5525.08	28.07	161.86	278.71	1529587
w = 1 *horiz.*	22.865	5579.43	28.01	164.31	271.14	1516267
w = 2/n *vert.*	23.839	5241.05	29.06	151.06	273.00	1413280
w = 2/n *horiz.*	24.275	5136.83	28.16	149.72	264.85	1361946
" " ("search" *smaller*)	23.960	5172.00	28.16	151.10	264.28	1372994
(Manual Placement)		5470.51	29.06	154.41	243.57	1328667

FIG. 27 - Results on the placement of AB6

produced results which are comparable to those obtained by manual placement. The best automatic solution was generated in approximatly 24 seconds (which includes the evaluation of objective functions and output of new net-list) and, for most of the assessed objectives, compares favourably with the manual placement solution.

The best solution obtained by the proposed method on the placement of board AB6 is represented in figure **28**. As in the connectivity tree in figure 26, ECONs are indicated by ○, XCONs by ● and the forbidden slots denoted by ⌀.

4. GENERAL CASE

A direct implementation of the tree-mapping process in its general form, as described in Ch IV. § 4, was tried on a number of PC boards, namely PC2, PC8 and PC11,all part of the MU6G hardware. The layout of these boards involved a set of interesting features such as dissimilar size of modules, edge connectors, preplacement and various degrees of component density.

For each board, different versions of the basic method were tested and the resulting solutions sent to the tracking system together with those obtained by manual placement. In the cases of boards PC8 and PC11 the results were similar and both comparable to those produced by the corresponding manual solutions. However, the placement of PC2 was difficult and not completely satisfactory. This fact was due to a very high density which, in the manual solution required a careful choice on the relative orientation of components. The set of orientations thus defined was naturally proved

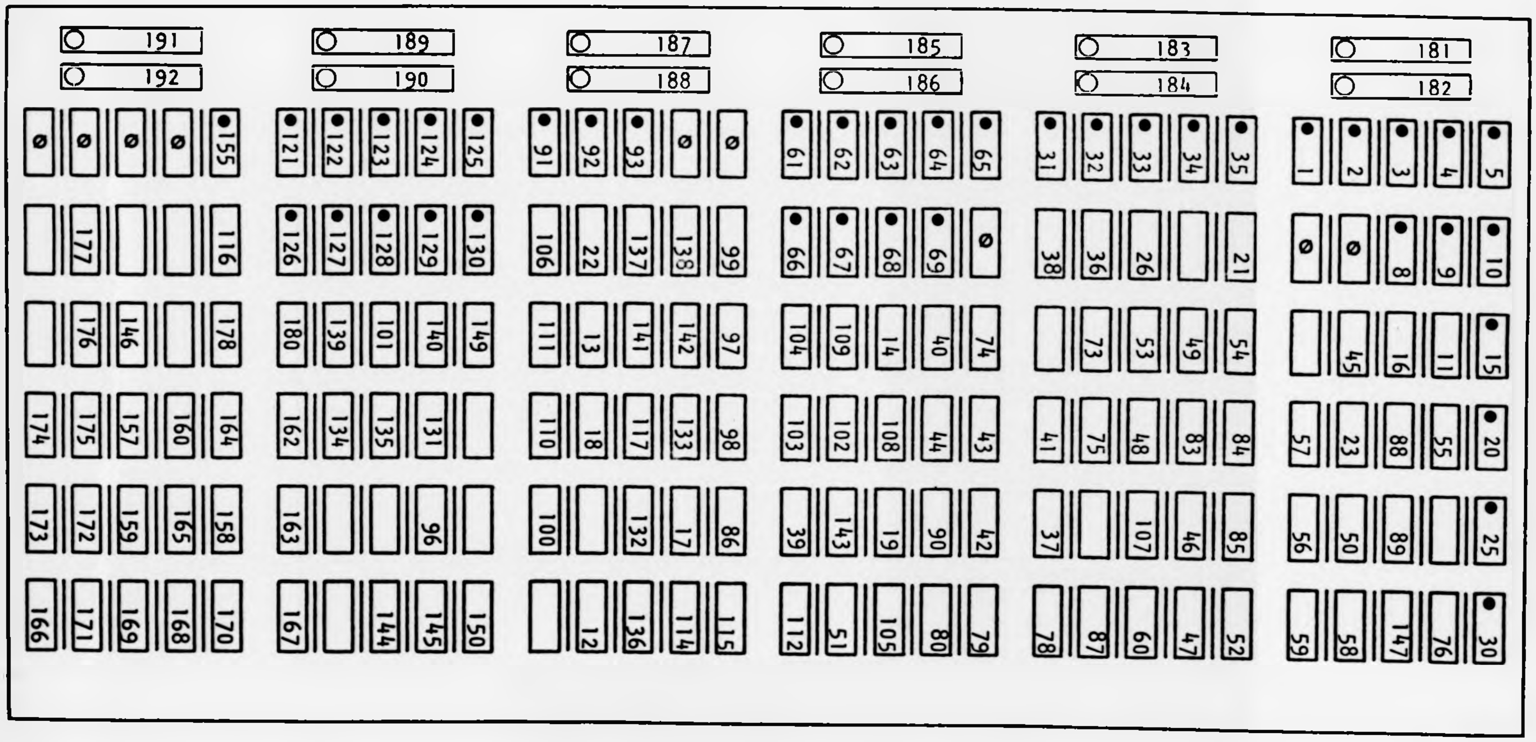


FIG. 28 - Placement solution of AB6

inadequate for different arrangements of modules such as those generated automatically. Since the practical implementation of the proposed placement method does not include the choice of component orientation, this problem had to be solve by manual intervention. The placement of PC2 was therefore only successfully accomplished after a sequence of empirical attempts.

In the remainder of this Chapter we will describe in detail the layout of PC8, a Memory board part of the MU6G. The circuit description of PC8 comprises a total of 125 modules as specified in the following table, connected by 485 nets.

	# modules	dimensions	area
16 - pin	80	7.5 x 20.0	12.5 x 25.0
22 - pin	36	10.0 x 28.0	15.0 x 33.0
ECONs	9	7.0 x 50.0	12.0 x 55.0

The indicated dimensions are all in millimetres and the assumed orientation for components is the one used in the manual solution: all modules placed vertically, excepting the edge connectors. As an approximation of the minimum necessary routing area, 5 millimetres were added to the real dimensions of modules. Since the board dimensions are 410 x 180, the ratio of the area occupied by components with respect to the total board area is approximately 2/3.

Also based on the manual placement solution, the 9 ECONs as well as 9 XCONs were preplaced on a 410 x 23.5 strip along the top board edge.

Two other XCONs were assigned to fixed positions within the remaining board area. These will be referred to as "preplaced" components and the former ones as "edge connectors".

Two versions of the basic method were tested, corresponding to the graph models with weight distributions $w = 1$ and $w = 2/n$. As already observed, the circuit representation with $w = 2/n$ generates a more unbalanced connectivity tree which requires a larger number of tree adjustments during the mapping phase.

4.1 - Tree adjustments

During the placement of board PC8, tree adjustments were performed in order to solve space conflicts and also to maintain the proportionality of areas, as discussed in Ch. IV. § 4.3. In fact, the implemented placement scheme failed to perform on this board without proportionality adjustment. We suspect that the density of components in PC8, i.e. 2/3 of the total board area, approaches the limit of successful performance of the implemented tree-mapping algorithm. A higher density may only be achieved by a more elaborated scheme including a packing algorithm or in association with a compaction routine.

Since the tree-mapping algorithm in its general form requires a larger number of operations per step, the observed running times were necessarily higher for the present type of boards. The following table reports the observed times in seconds of the fundamental phases in the placement of PC8, for both graph models, and the corresponding values of the total weighted Euclidean distance and maximum single length in millimetres.

	tree-building	tree-mapping	dist	max
w = 1	7.652	5.956	6.587×10^4	3.256×10^2
w = 2/n	7.507	7.508	6.527×10^4	3.848×10^2
(Manual Placement)			5.938×10^4	3.060×10^2

As already pointed out, the representation with w = 2/n generates a more unbalanced tree, which increases the number of tree adjustments during the mapping process. In the remainder of this paragraph we will analyse the effect of such adjustments on the running time of the tree--mapping procedure, using the connectivity tree obtained on the graph model with w = 1 as example, for reasons of simplicity.

Figure 29 illustrates the basic tree for board PC8 with weights distribution w = 1. Edge connectors are denoted by ○ (ECONs) and ● (XCONs). Preplaced components E4 and E5 are indicated by □. This tree is subsequently modified during the mapping phase, due to three main factors: space conflicts, proportionality adjustments and preplacement. The resulting placement solution is depicted in figure 30.

Figure 31 shows the aspect of the same connectivity tree, after the mapping process execution. In this diagram, terminal branches are truncated whenever they comprise an easily identifiable block of components and edge connectors are not indicated, although included in the counting at each tree vertex. Manifest differences between the trees are the adjustment of "long branches" in order to fit board areas and the "displacement" of preplaced components E4 and E5 (□).

The values in brackets at each tree vertex, denote the observed running times of the corresponding mapping steps. These values are represented

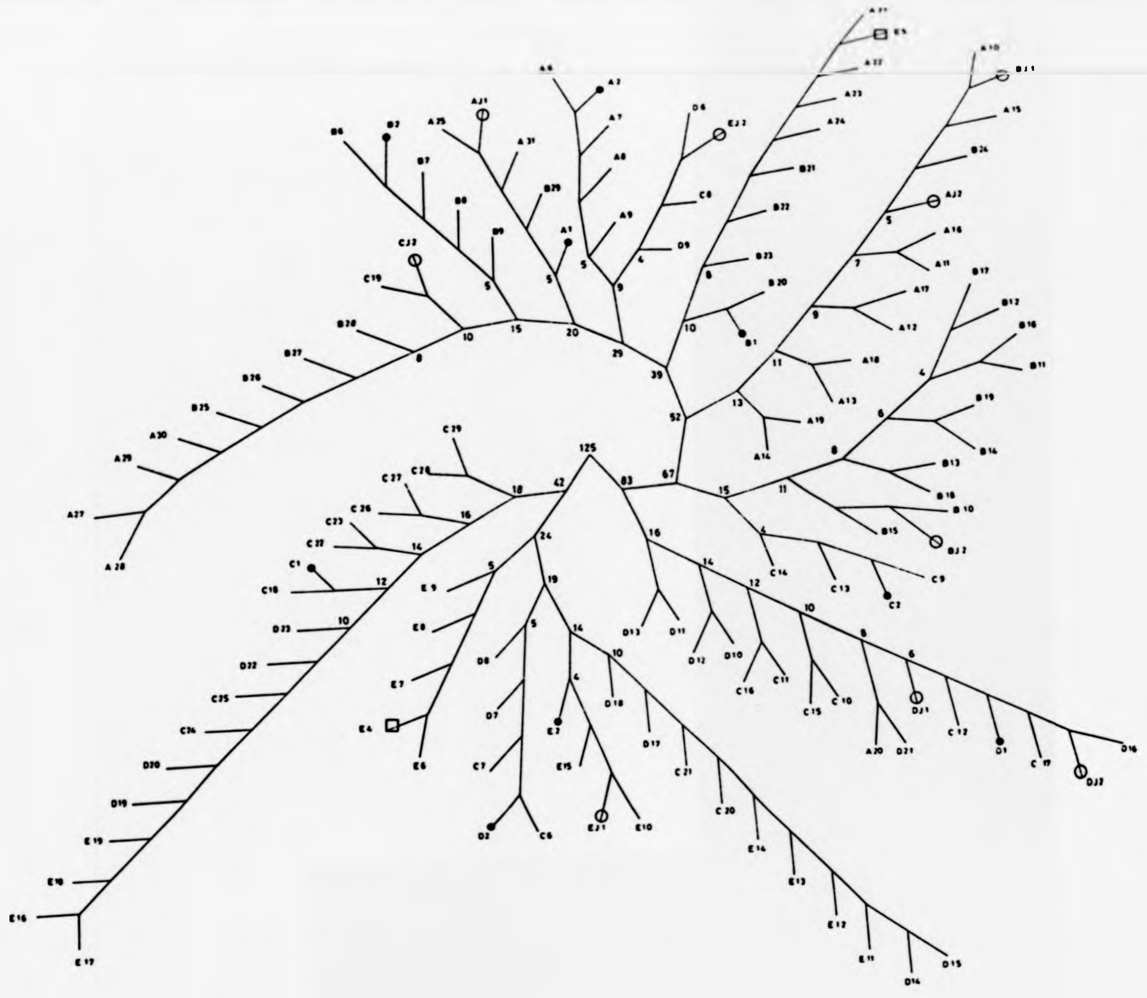


FIG. 29 - Connectivity tree for PC8 with w = 1

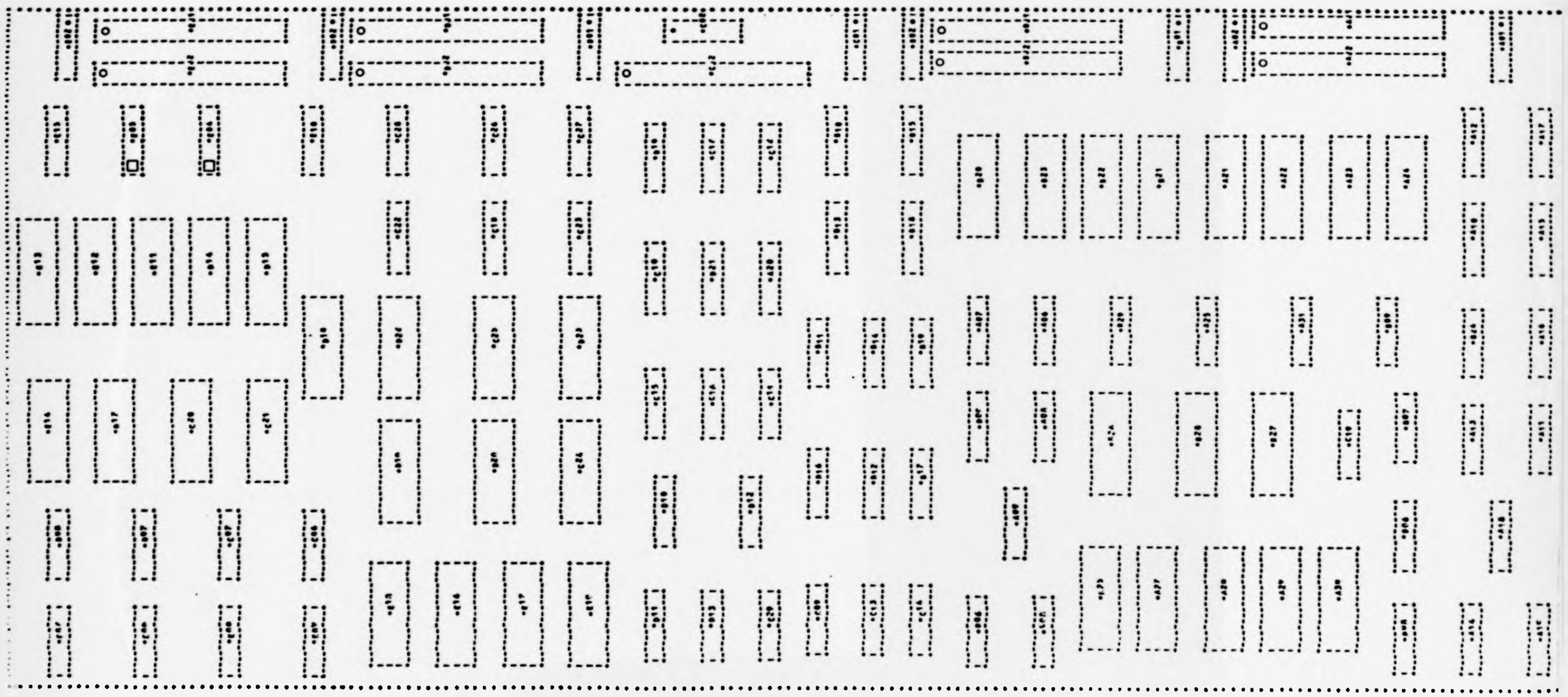


FIG. 30 - Automatic placement solution of PC8 with w = 1

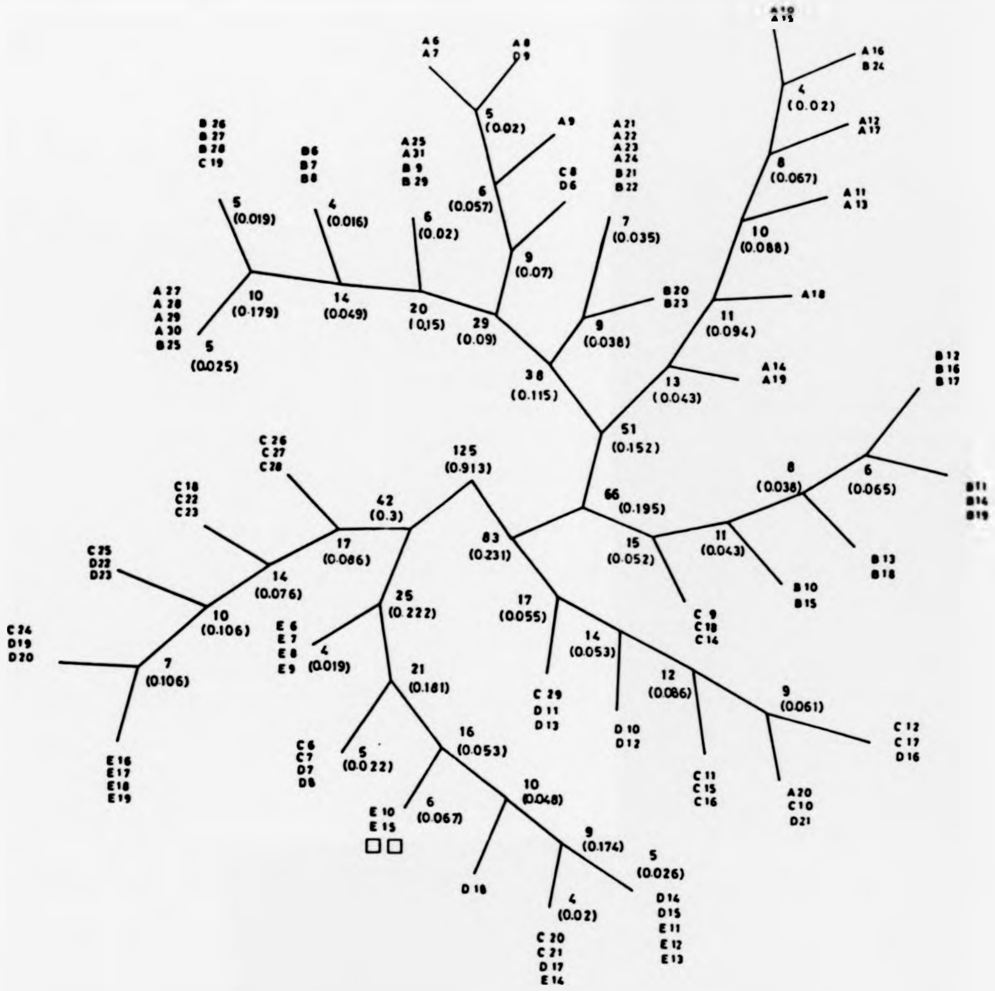


FIG. 31 - Connectivity tree for PC8 (w=1), after mapping

as points in the graphic at figure **32**, where the x-coordinate indicates the number of components at each vertex and the y-coordinate is the observed running time in seconds. As pointed out in Ch IV . § 4.4, the process is near-linear at the steps where no alteration is performed. Higher time values correspond to those steps where elements have been transferred from one sub-branch to the other. As an example, at the tree root one preplaced module (1P) was transferred from the right to the left hand side and component C29 was displaced (1d) in the opposite direction. The point at coordinates (10, 0.179) shows that 3 displacements (3d) were required in order to fit a "long branch" into an approximatly square board area. Note that tree adjustments due to space conflict are more frequent for smaller knots, representing lower tree vertices.

Since the computational complexity for the traversal of a perfectly balanced binary tree is of order $O(n \log_2 n)$ and based on the observed running times, we may estimate the minimum time required by the tree-mapping process. In the ideal case where no adjustments are performed and assuming a symmetrical connectivity tree with $\underline{n}$ leaves, the mapping phase will take at least:

$$T_{map}(n) = \alpha . n . \log_2 n \qquad \text{with} \quad \alpha \approx 0.003 \text{ sec.}$$

4.2 - Tracking

Both placement solutions of board PC8 (based on the graph models with $w = 1$ and $w = 2/n$) were tried on the existing tracking system, together with the manual layout. The observed rates of successful connections, although not completely satisfactory, showed that the method can produce

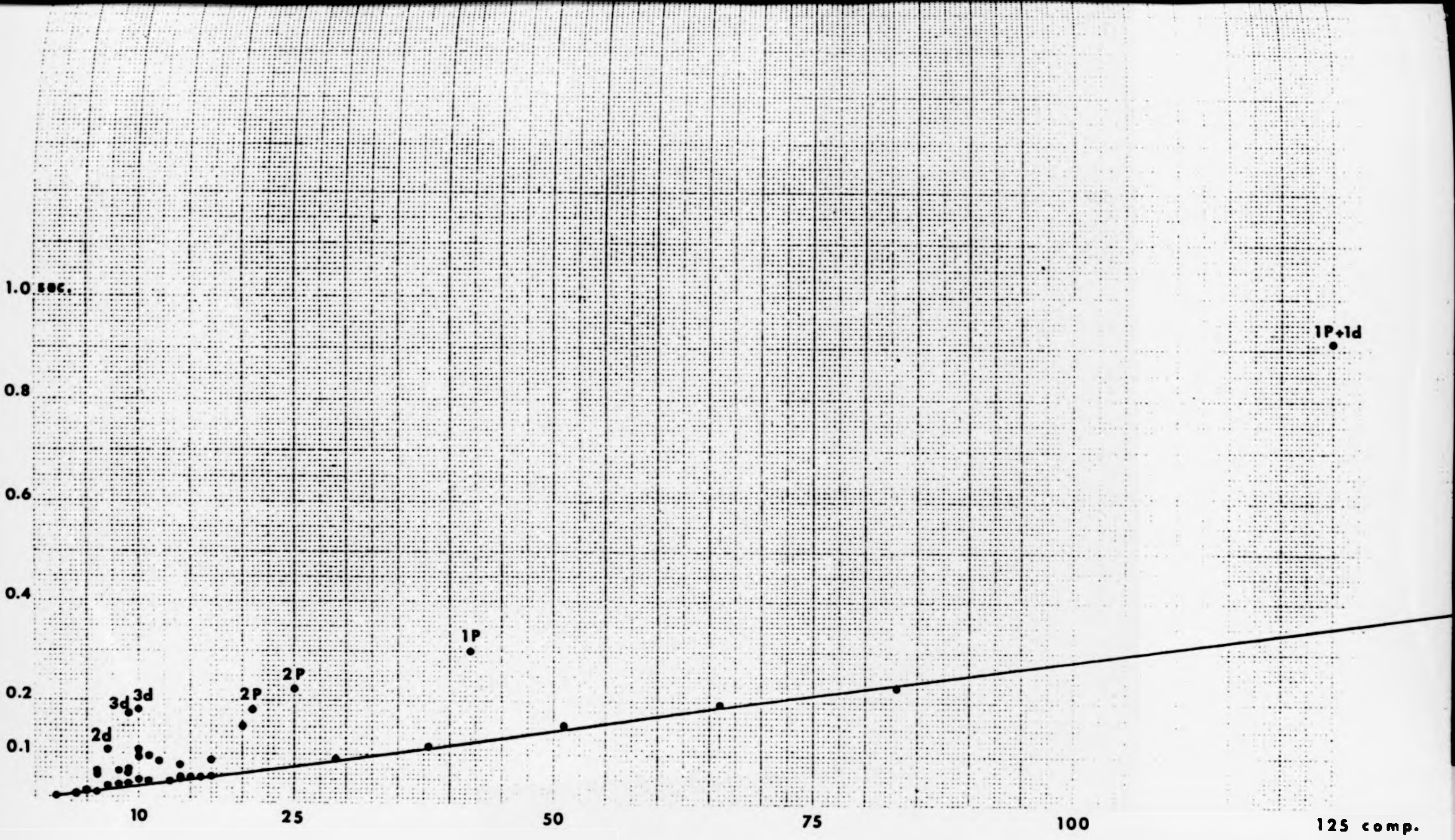


FIG. 32 - Running times at each step of the tree-mapping process

solutions which are comparable to those obtained manually. Routing results were also used in order to assess the relative efficiency of different versions of the basic method and to analyse the effect of local adjustments on the final layout.

From 1693 initial points, 52.86% were successfully connected on the automatic solution with w = 1 and only 45.12% on the one with w = 2/n, whereas 52.51% were connected on the manual placement. It should however be pointed out that the described implementation did not include any form of adjustment in terms of the available list of connections. Component orientation and pin assignment were the same as for the manual solution and no reordering was performed either within the elements of each net or on the given net sequence.

Figures 33 and 34 show the successful connections generated by the tracking system over the placement solution of board PC8 based on the graph model with w = 1. Although with similar total Euclidean distance (see § 4.1) the solution with w = 2/n produced manifestly lower routing results. We suspect that this is mainly due to the larger number of transferred elements which causes slight misalignments between pins of adjacent components. Recall that in the implementation described, modules are placed at the geometric centre of the corresponding rectangular domain. On the other hand, the exact dimensions of such domains may easily be made available to the user at the end of the mapping process. This suggests that pin-alignment can be performed in a simple post-processor which may considerably enhance the routability of a placement solution. In the present example 26 modules were selected as noticeably "displaced" and accordingly aligned with neighbouring components. As a result, 94 new points were successfully connected, which improved the observed rate from 45.12% to 50.68%.

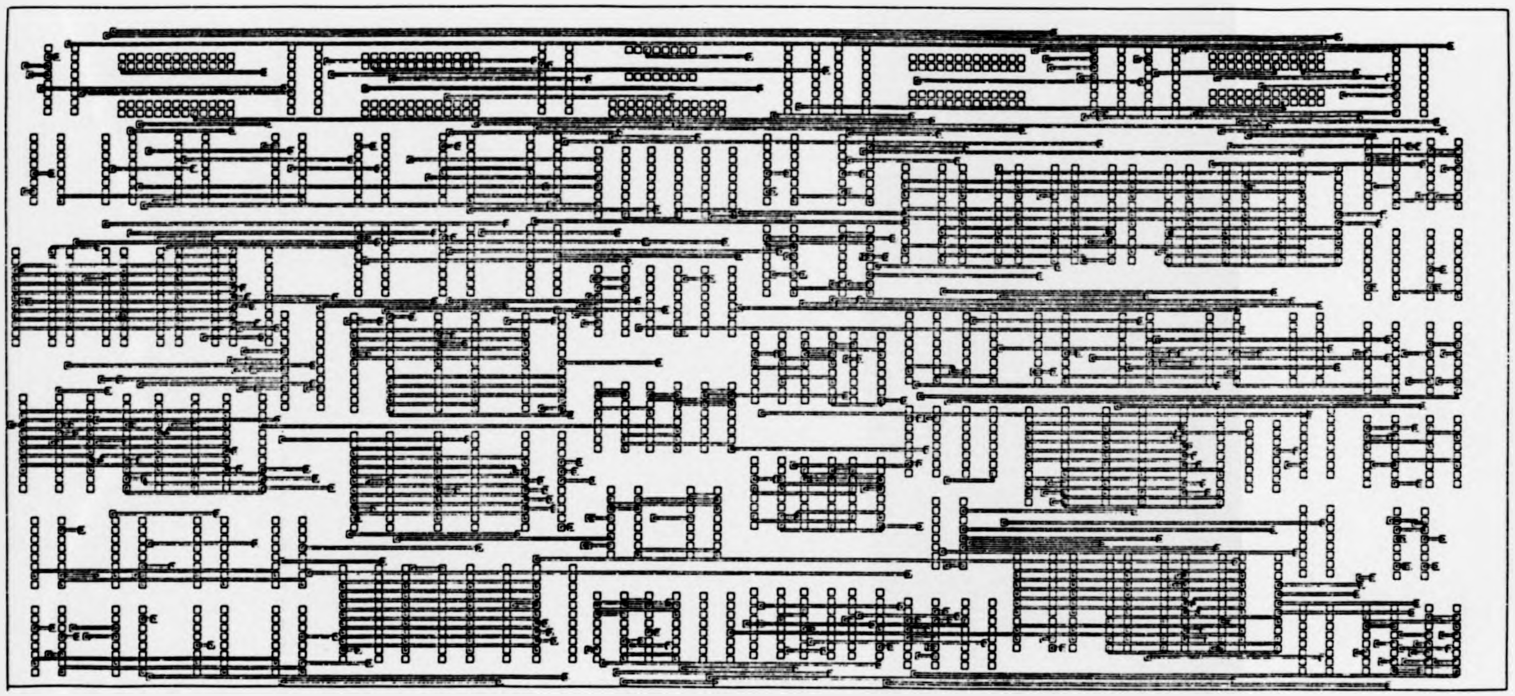


FIG. 33 - Plotted wire pattern of PC8 (w =1): horizontal connections

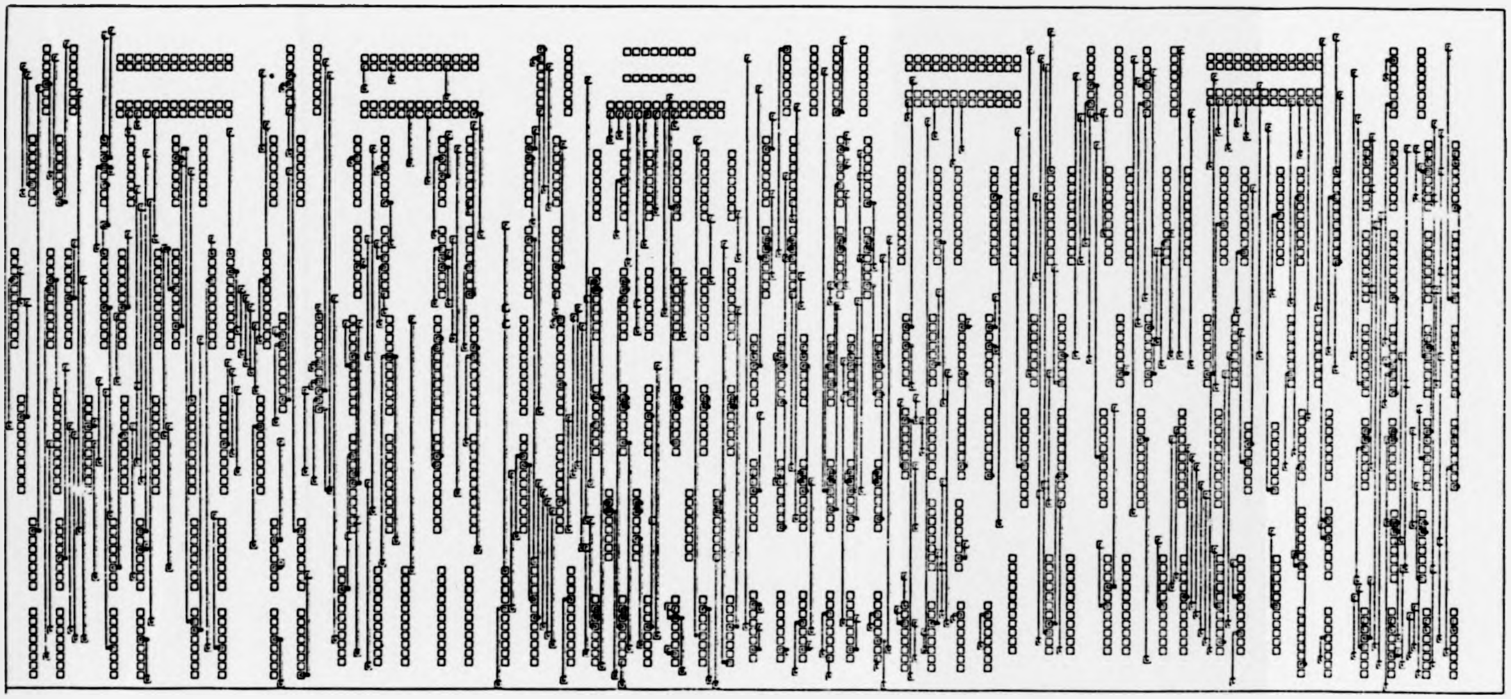


FIG. 34 - Plotted wire pattern of PC8 (w=1): vertical connections

CHAPTER VI

CONCLUSIONS

1. A REVIEW OF CONCEPTS

The intrinsic complexity of the *placement problem*, together with the diversity of layout environments, has led to the development of a wide range of placement techniques which are diverse in their formulation, aims and strategy. In this thesis we presented a *definition* of the placement problem in mathematical terms and an attempt was made in order to systematize its most widely accepted figures of merit. For this effect a number of *objective functions* was defined, which are independent of any particular technology or routing scheme and can thus be evaluated directly from the obtained placement configurations. Three basic approaches to the problem were specified and accordingly a *classification* of the existing placement methods was proposed.

The layout of integrated circuits motivated the use of *hierarchical methods* as a means of dealing with increasing circuit complexity. However this approach has not been, in our belief, completely explored. In most of the proposed techniques, the hierarchy of blocks is either defined by the user and based on functionality [PG78] [PV79] or determined by an initial placement configuration obtained with the aid of one of the classical methods [MS81] [BK83] [OI85]. Few attempts have been made to build up the hierarchical structure first and then to map it into the board surface. In this case, the tree structure is generated by a clustering algorithm which aims strictly to minimize the total wire distance [MT79] [Ha82] [Ri84].

In this thesis we suggested a new approach to hirarchical placement, based on a *tree structure which embodies an adopted set of circuit properties and objective functions*. The tree structure thus defined represents the relative positions of a hierarchy of blocks on the board surface. Under this approach, no initial configuration is required and no iterative interchange is performed either during the building or the embedding of the tree. The proposed strategy combines the simplicity of constructive placement techniques with the global circuit overview provided by classical methods as well as the expedience of a fully hierarchical approach.

We believe that efficient algorithms can be designed,both for the building of such a tree structure and its subsequent embedding on the different types of layout environments. In this thesis, a new *placement method* has been described which was designed in accordance with the proposed philosophy. It comprises a tree-building algorithm based on the optimization of three placement objectives and two algorithms for the tree-mapping on the basic types of boards: regularly structured and continuous plane.

The initial phase of the proposed method consists of representing the given physical circuit by a suitable *graph model*. A simple model was adopted which can represent the basic topological circuit properties as well as other types of design requirements.

The graph model acts as a basis for the building of the tree structure. In Chapter III we introduced a number of graph theoretical concepts, which enabled a formulation of the correspondence between the structure of a binary tree and desirable placement properties. The optimal *connectivity tree* thus defined represents the relative positions of blocks in a placement solution with minimal total distance and even distribution of both wires and components on the board surface. However, these objectives represent complex combinatorial problems and are, is some cases, incompatible. For that reason an approximation *algorithm* was proposed, which tries to meet the three objectives as closely as possible while being competitive in terms of running time and memory space.

In Chapter IV we described the *mapping of the connectivity tree* into the given board area. Two algorithms were presented, one for *regularly structured boards* with fixed locations for modules and the other for the *general case* of a continuous plane where components of dissimilar size may occupy any distinct positions. The algorithms are compatible with specific design requirements such as the preplacement of components at definite coordinates and, to some extent, adjust the disposition of components in order to fit densely occupied areas. In the general placement case the method also provides, at each tree vertex, an estimation of the corresponding layout area including the interconnect space.

A *binary structure* was adopted for the connectivity tree in order to simplify the tree-mapping process which would otherwise lead to the complex problem of non-guillotine rectangle cutting. The main desidvantage of a binary approach lies in the fact that it cannot, by itself, generate a placement configuration with the smallest possible area. However, a placement solution which accomplishes the adopted set of circuit objectives can be efficiently generated by a binary scheme and *subsequently compacted* into as small an area as possible, in the cases where a minimal area is required. The recent advances in the field of circuit compaction [Ch85] came to confirm this assumption, by providing a choice of compaction methods which are compatible with the basic binary scheme.

The placement method proposed in this thesis was implemented in Pascal on a CDC 7600 system and tried on a number of PC boards of the two basic types: regularly structured and continuous plane. The observed values of objective functions and running times confirmed the estimated efficiency and complexity of the algorithms involved. A comparison with manually obtained placement configurations showed that the basic method can produce solutions which are competitive both in terms of measurable properties and observed routability.

2. SUGGESTIONS FOR FURTHER RESEARCH

A fully hierarchical placement philosophy has been suggested in this thesis and a new placement method has been designed and implemented as a prototype of that approach. The observed results indicate that the method is fast and produces placement configurations which are comparable with those obtained manually. This fact attests the expedience of the proposed philosophy and provides motivation for further research.

The goodness of a placement is ultimately determined by how efficiently it can be routed. The experience of human designers has shown that placement objectives such as minimal distance and even distribution are strongly related to circuit routability. However, there exist a number of empirical factors which are taken into account by skillful designers and have not yet been formulated in mathematical terms. Manually obtained placement configurations are, for that reason and at least for small problems, usually more suitable than the automatically generated ones. We suggest that a complete study be made on the circuit properties which are known to increase routability for each particular type of layout. Once those properties are formalized in mathematical terms, we assume that effective algorithms can be designed which are specifically suited to each problem. A placement system could then provide a choice of both tree-mapping and tree-building algorithms to be selected for each particular layout problem.

An example of the referred empirical factors appears to be the identification of highways and bus-structured nets as well as their subsequent positioning on the circuit board. We suspect that such structures, once identified, can be treated as new vertices to be coalesced during the tree--building process.

Two algorithms have been proposed for the tree-mapping process and tried on a number of PC boards. Although not confirmed by practical implementation, the first algorithm is also applicable to other types of regularly structured environments such as gate arrays and cellular arrays. The second algorithm contemplates the general case of a continuous plane whenever a minimal circuit area is not required, as discussed in § 1 of this Chapter. The method can only be made applicable to layout problems such as the design of *full-custom VLSI chips,* when used in association with a compaction procedure. This approach is also being followed by an increasing number of layout automation systems which deal with custom IC design [Hs81] [RM83] [LJ84] [NJ85] [RR85].

The *constraint graph approach to compaction*, as outlined in Chapter II. § 4.4, is particularly adequated to a binary space-partitioning process. The constraint graphs can be generated during the tree-mapping phase in a manner which is similiar to the polar graphs construction (Ch II. § 4.2). An estimation of minimal circuit dimensions is given by a longest path evaluation in each graph. Further minimization of the total area can still be achieved by local adjustments of particular components. We suggest that a compaction method can be developed where components situated along the critical path of each graph are *rotated* whenever this operation does not increase the total chip area. This approach could require several iterations and repeated longest path evaluations but appears viable be means of network programming techniques.

The fully hierarchical placement philosophy proposed in this thesis is compatible with a wide range of layout environments. This fact enables the use of the same method at various levels of the tree structure representing the system hierarchy. The connectivity tree of a given circuit thus becomes a branch in the physical hierarchy of the whole system. Under this perspective, we believe it is viable to apply the basic philosophy to other aspects of the layout problem. As an example, a tree-building algorithm could be applied to the layout processes which involve partitioning.

The connectivity tree also provides valuable information to be used during the routing phase. We suggest that the sequence in which the wires are processed, could be defined in terms of a bottom-up tree ordering. This approach is compatible with hierarchical routing methods which use a bottom--up and shortest-first scheme [LS80] [BP83] [RM83].

It could also be interesting to investigate the suitability of the approach to an effective interaction between the placement and the routing. An integrated layout system could then be developed where, at different levels of the tree structure, the placement is performed, pin positioning is optimized and the partially formed configuration is routed and compacted.

REFERENCES

[AG70] AKERS, S.B.; GEYER, J.M. and ROBERTS, D.L.
"IC Mask Layout with a Single Conductor Layer"
Proceedings of the 7th Designe Automation Workshop, 1970, pp. 7-16.

[AH74] AHO, A.V.; HOPCROFT, J.E. and ULLMAN, J.D.
"The Design and Analysis of Computer Algorithms"
Addison-Wesley, Massachusetts, 1974.

[AK71] ARAMAKI, I.; KAWABATA, T. and AKIMOTO, K.
"Automation of Etching-Pattern Layout"
Communications of the ACM, Vol. 14, No.11, Nov. 1971, pp. 720-730.

[Be70] BERGE, C.
"Graphes et Hypergraphes"
Monographies Universitaires de Mathématiques, Dunod, Paris 1970.

[BK83] BREUER, M.A. and KUMAR, A.
"A Methodology for Custom VLSI Layout"
IEEE Trans. on Circuits and Systems, Vol. CAS-30, No. 6, June 1983, pp. 358-364.

[Bl85] BLANKS, J.P.
"Near-Optimal Placement Using a Quadratic Objective Function"
Proceedings of the 22nd Designe Automation Conference, 1985, Paper 38.2, pp. 609-615.

[BP83] BURSTEIN, M. and PELAVIN, R.
"Hierarchical Wire Routing"
IEEE Trans. on COMPUTER-AIDED DESIGN, Vol. CAD-2, No. 4, October 1983, pp. 223-234.

[Br77] BREUER, M.A.
"A Class of Min-Cut Placement Algorithms"
Proceedings of the 14th Design Automation Conference, 1977, pp. 284-290.

[CB82] CHANDRASEKHAR, M.S. and BREUER, M.A.
"Optimum Placement of Two Rectangular Blocks"
Proceedings of the 19th Design Automation Conference, 1982, pp. 879-886.

[Ch85] CHO, Y.E.
"A Subjective Review of Compaction"
Proceedings of the 22nd Design Automation Conference, 1985, Paper 25.1, pp. 396-404.

[CK84] CHENG, C.K. and KUH, E.S.
"Module Placement Based on Resistive Network Optimization"
IEEE Trans. on COMPUTER-AIDED DESIGN, Vol. CAD-3, No. 3, July 1984, pp. 218-225.

[CM73] CROCKER, N.R.; McGUFFIN, R.W.; NAYLOR, R. and VOSPER, H.
"Computer Aided Placement and Routing of High Density Chip Interconnection Systems"
Proc. Conf. on COMPUTER AIDED DESIGN FOR ELECTRONIC CIRCUITS,AGARD, NATO, 1973, Paper 32, pp. 1-26.

[Co79] CORRIGAN, L.I.
"A Placement Capability Based on Partitioning"
Proceedings of the 16th Design Automation Conference, 1979, pp. 406-413.

[FF62] FORD, L.R. and FULKERSON, D.R.
"Flows in Networks"
Princeton, New Jersey, Princeton University Press, 1962, p. 11.

[Ga81] GALLE, P.
"An Algorithm for Exhaustive Generation of Building Floor Plans"
Communications of the ACM, Vol. 24, No. 12, December 1981, pp.813-825.

[GJ74] GAREY, M.R.; JOHNSON, D.S. and STOCKMEYER, L.
"Some Simplified NP-complete Problems"
Proceedings of the 6th Annual ACM Symposium of the Theory of Computing"
Seattle, April 1974, pp. 47-63.

[GJ81] GAREY, M.R. and JOHNSON, D.S.
"Approximation Algorithms for Bin-packing Problems: A Survey"
in Analysis and Design of Algorithms in Combinatorial Optimization, G. Ausiello and M. Lucertini, eds., Springer-Verlag, New York, 1981, pp. 147-172.

[GJ83] GAREY, M.R. and JOHNSON, D.S.
"Crossing Number is NP-complete"
SIAM Journal of Alg. Disc. Meth., Vol. 4, No. 3, Sept. 1983, pp. 312-316.

[GK85] GLOVER, F.; KLINGMAN, D.D.; PHILLIPS, N.V. and SCHENEIDER, R.F.
"New Polynomial Shortest Path Algorithms and their Computational Attributes"
Management Science, Vol. 31, No. 9, Sept. 1985, pp. 1106-1128.

[Ha82] HASSETT, J.E.
"Automated Layout in ASHLAR: An Approach to the Problems of 'General Cell' Layout for VLSI"
Proceedings of the 19th Design Automation Conference, 1982, pp. 777-784.

[HG85] HEALEY, S.T. and GAJSKI, D.D.
"Decomposition of Logic Networks into Silicon"
Proceedings of the 22nd Design Automation Conference, 1985, Paper 13.1, pp. 162-168.

[HK72] HANAN, M. and KURTZBERG, J.M.
"Placement Techniques"
in 'Design Automation of Digital Systems, Theory and Techniques',
Vol. 1, Chap. 5, Prentice-Hall, New Jersey, 1972.

[Hs81] HSUEH, M.Y.
"Symbolic Layout Compaction"
in 'Computer Design Aids for VLSI Circuits'
P. Antognetti, D.O. Pederson and H. de Man, eds., Nato Advanced Study Institutes Series, SERIES E: Applied Sciences - No. 48, 1981,
pp. 499-541.

[HS82] HELLER, W.R.; SORKIN, G. and MALING, K.
"The Planar Package Planner for System Designers"
Proceeding of the 19th Design Automation Conference, 1982,
pp. 253-260.

[HT74] HOPCROFT, J.E. and TARJAN, R.E.
"Efficient Planarity Testing"
Journal of the ACM, Vol. 21, 1974, pp. 549-568.

[HW76] HANAN, M.; WOLF, P.K. and AGULE, B.J.
"Some Experimental Results on Placement Techniques"
Proceedings of the 13th Design Automation Conference, 1976,
pp. 214-224.

[HW84] HUDSON, J.A.; WISNIEWSKI, J.A. and PETERS, R.C.
"Module Positioning Algorithms for Rectilinear Macrocell Assemblies"
Proceeding of the 21st Design Automation Conference, 1984,
Paper 45.2, pp. 672-675.

[KL70] KERNIGHAN, B.W. and LIN, S.
"An Efficient Heuristic Procedure for Partitioning Graphs"
The Bell System Technical Journal, Vol. 49, Feb. 1970, pp. 291-307.

[La73] LAWLER, E.L.
"Cutsets and Partitions of Hypergraphs"
Networks, Vol. 3, No. 3, 1973, pp. 275-285.

[La79] LAUTHER, U.
"A Min-Cut Placement Algorithm of General Cell Assemblies Based on a Graph Representation"
Proceedings of the 16th Design Automation Conference, 1979, pp. 1-10.

[Li77] LINDSEY, C.H.
"Structure Charts: A Structured Alternative to Flowcharts"
Dept. of Computer Science, University of Manchester, Internal Report, 1977.

[LJ84] LOTVIN, M.; JURAN, B. and GOLDIN, R.
"AMOEBA: A Symbolic VLSI Layout System"
Proceedings of the 21st. Design Automation Conference, 1984,
Paper 19.2, pp. 294-300.

[LL69] LAWER, E.L.; LEVITT, K.N. and TURNER, J.
"Module Clustering to Minimize Delay in Digital Networks"
IEEE Trans. on Computers,Vol. C-18, No. 1, Jan. 1969, pp. 47-57.

[Lo84] LOYNS, A.
"Generalised Tracking System: Program Documentation"
Dept. of Computer Science, University of Manchester, Internal Report, Dec. 1984.

[LS69] LUCCIO, F. and SAMI, M.
"On the Decomposition of Networks in Minimally Interconnected Subnetworks"
IEEE Trans. in Circuit Theory, Vol. CT-16, No. 2, May 1969, pp. 184-188.

[LS80] LORENZETTI, M.J. and SMITH II, R.J.
"An Implementation of a Saturated Zone Multi-Layer Printed Circuit Board Router"
Proceedings of the 17th Design Automation Conference, 1980, pp. 255-262.

[MC80] MEAD, C. and CONWAY, L.
"Introduction to VLSI Systems"
Addison-Wesley Publishing Company, Massachusetts, 1980.

[MJ84] MARKOV, L.A.; FOX, J.R. and BLANK, J.H.
"Optimazation Techniques for Two-Dimensional Placement"
Proceedings of the 21st. Design Automation Conference, 1984, Paper 44.1, pp. 652-654.

[MM82] MALING, K.; MUELLER, S.H. and HELLER, W.R.
"On Finding Most Optimal Rectangular Package Plans"
Proceedings of the 19th Design Automation Conference, 1982, pp. 663-670.

[MS81] MALLADI, R.; SERRERO, G. and VERDILLON, A.
"Automatic Placement of Rectangular Blocks with the Interconnection Channels"
Proceedings of the 18th Design Automation Conference, 1981, pp. 419-425.

[MT79] MURAI, S.; TSUJI, H.; KAKINUMA, M.; SAKAGUCHI, K. and TANAKA, C.
"A Hierarchical Placement Procedure with a Single Blocking Scheme"
Proceedings of the 16th Design Automation Conference, 1979, pp. 18-23.

[NJ85] NG, T.K. and JOHNSSON, S.L.
"Generation of Layouts from MOS Circuit Schematics: a Graph Theoretic Approach"
Proceedings of the 22nd Design Automation Conference, 1985, Paper 4.2, pp. 39-45.

[OI85] ODAWARA, G.; IIJIMA, K. and WAKABAYASHI, K.
"Knowldge-Based Placement Technique for Printed Wiring Boards"
Proceedings of the 22nd Design Automation Conference, 1985,
Paper 38.3, pp. 616-622.

[PG78] PREAS, B.T. and GWYN, C.W.
"Methods for Hierarchical Automatic Layout of Custom LSI Circuit Masks"
Proceedings of the 15th Design Automation Conference, 1978,
pp. 206-212.

[PV79] PREAS, B.T. and vanCLEEMPUT, W.M.
"Placement Algorithms for Arbitrarily Shaped Blocks"
Proceedings of the 16th Design Automation Conference, 1979,
pp. 474-480.

[QB79] QUINN Jr., N.R.; and BREUER, M.A.
"A Force Directed Component Placement Procedure for Printed Circuit Boards"
IEEE Trans. Circuits Systems, Vol. CAS-26, June, 1979, pp. 377-388.

[Ri84] RICHARD, B.D .
"A Standard Cell Initial Placement Strategy"
Proceedings of the 21st Design Automation Conference, 1984,
Paper 25.2, pp. 392-398.

[RM83] ROTHERMEL, H.J. and MLYNSKI, D.A.
"Automatic Variable-Width Routing for VLSI"
IEEE Trans. on COMPUTER-AIDED DESIGN, Vol. CAD-2, No. 4, October 1983, pp. 271-284.

[RR85] ROGERS, C.D.; ROSENBERG, J.B. and DANIEL, S.W.
" MCNC's Vertically Integrated Symbolic Design System"
Proceedings of the 22nd Design Automation Conference, 1985,
Paper 5.1, pp. 62-68.

[SD85] SHA, L. and DUTTON, R.W.
"An Analytical Algorithm for Placement of Arbitrarily Sized Rectangular Blocks"
Proceedings of the 22nd Design Automation Conference, 1985,
Paper 38.1, pp. 602-608.

[SK72] SCHWEIKERT, D.G. and KERNIGHAM, B.W.
"A Proper Model for the Partitioning of Electrical Circuits"
Proceedings of the 9th Design Automation Workshop, 1972, pp. 57-62.

[St61] STEINBERG, L.
"The Backboard Wiring Problem: a Placement Algorithm"
SIAM Review, Vol. 3, No. 1, January 1961, pp. 37-50.

[Va76] VanCLEEMPUT, W.M.
"Mathematical Models and Algorithms for the Circuit Layout Problem"
Ph. D. Thesis, University of Waterloo, June 1976.

[Wa80] WANG, P.T.
"A Placement Technique Based on Minimization and Even Distribution of Crossovers"
SIGDA Newsletter, Vol. 10, No. 3, August 1980, pp. 9-20.

[Wi76] WIRTH, N.
"Algorithms + Data Structures = Programs"
Prentice-Hall, Series in Automatic Computation, New Jersey, 1976.

[Wi78] WILLIAMS, J.D.
"STICKS - A Graphical compiler for High Level LSI Design"
AFIPS Conf. Proc., Vol. 47, June 1978, pp. 289-295.

[WW82] WIPFLER, G.J.; WIESEL, M. and MLYINSKI, D.A.
"A Combined Force and Cut Algorithm for Hierarchical VLSI Layout"
Proceedings of the 19th Design Automation Conference, 1982,
pp. 671-677.