MATHEMATICAL MODELS AND ALGORITHMS FOR THE CIRCUIT LAYOUT PROBLEM

by

W.M. van Cleemput

Research Report CS-76-29

Department of Computer Science

University of Waterloo
Waterloo, Ontario, Canada

June 1976

MATHEMATICAL MODELS AND ALGORITHMS FOR THE CIRCUIT LAYOUT PROBLEM

by

W.M. vanCleemput

A Thesis
Submitted in partial fulfillment of
the requirements for the degree of
DOCTOR OF PHILOSOPHY

at the
University of Waterloo
Waterloo, Ontario, Canada
Department of Computer Science
August 1975

ABSTRACT

The research reported in this thesis deals with the problem of circuit layout. Rather than partitioning the problem into placement and routing phases, graph-theoretical methods will be studied. The methods are based mainly on the concepts of planarity and of embedding graphs in the plane.

The first part of this dissertation deals with mathematical models for the circuit layout problem. As a result, new and greatly improved graph-theoretic models for the circuit layout problem are developed.

In the second part, a number of algorithms will be presented, dealing with testing the planarity of oriented graphs, the embedding of a maximal planar subgraph of a circuit layout graph and with using the technological properties of the problem for completing the layout. These algorithms are applicable especially for the layout of integrated circuits.

ACKNOWLEDGEMENTS

The author is especially grateful to his supervisor, Professor J.G. Linders, for his guidance and suggestions during the research, leading to this dissertation.

The criticism and suggestions of W. Brehaut, C. Lucchesi and J.A. Smith were invaluable.

The financial support of the National Research Council of Canada is gratefully aknowledged.

Last, but not least, the author thanks his wife Magda for her patience and support.

Table of Contents.

0. Introduction.

The circuit layout problem is an important facet of design automation of digital systems. The problem is encountered in laying out printed circuit boards, integrated circuit masks, logic diagrams, flowcharts, electronic circuit diagrams, etc.

The research reported in this thesis was motivated by two major facts: first of all, a satisfactory mathematical formulation of the problem seems to be lacking; second, certain types of layout problems, especially the layout of large scale integrated circuits, have proven to be difficult to solve optimally by using the conventional approach, used mainly for printed circuit layout.

Consequently, the thesis can be divided into two major parts: in the first three chapters, mathematical models for the problem will be described, while the remaining chapters are devoted to algorithms and procedures that can be applied to solve a particular problem. The emphasis in this thesis is specifically towards the mathematical models for the solution of the circuit layout problem.

In chapter 1, the circuit layout problem is described. and the main properties of the classical and of the topological layout methods are highlighted. A critical discussion of mathematical models, used for the layout problem, as found in the current litterature, is given.

In the second chapter, an improved graph-theoretic model for the problem is presented. Besides correcting some of the problems, associated with the models described in chapter 1, it allows one to perform pin and gate assignment as a function of the layout. In chapter 3, the mathematical properties of physical circuits are carefully scrutinized. Based on these properties, a mathematical model is derived for components, nets and outside connections. This model is a generalization of the one presented in chapter 2.

After a graph model is built for the circuit, it is necessary to embed this circuit graph or a maximal subgraph thereof in one or more layers (planes). Chapter 4 first describes some existing algorithms and their problems with respect to embedding circuit graphs of the kind derived in chapter 3. In particular, an efficient algorithm is presented for solving the problem.

Even if the graph model is not planar, it can frequently be planarized by making use of technological properties of the particular problem. Some of these properties and algorithms for using them are presented in chapter 5.

The algorithms, referenced so far, allow one to obtain a topological embedding of the circuit layout graph i.e. a list of oriented faces of the graph. In chapter 5, an algorithm will be described for deriving a preliminary physical (geometrical) embedding of the circuit layout graph. †

---

† Graph theoretic definitions are given mainly in sections 3.0 and 4.1. Terms that are not defined follow Harary (Ha69).

1. The Use of Topological Models for the Circuit Layout Problem - a Survey.

1.1 The Circuit Layout Problem.

The problem of laying out a circuit 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 into account a number of constraints. These constraints depend on the specific circuit layout problem being considered.

A well known circuit layout problem is the generation of electronic circuit drawings. Here the components are placed in a plane and the connections are realized in the same plane. Connections are allowed to cross each other. The parameters to be optimized are related to human perception and are rather difficult to measure. One could try to minimize the number of crossings of connections and/or the physical distance between logically related (i.e. connected) components.

The layout of logic diagrams is a similar problem. However, here the components have the property that that some of their pins are equivalent (i.e. mutually interchangeable). This property allows the elimination of crossings between connections without changing the logical structure. As an example, consider Fig. 1.1.1 where two

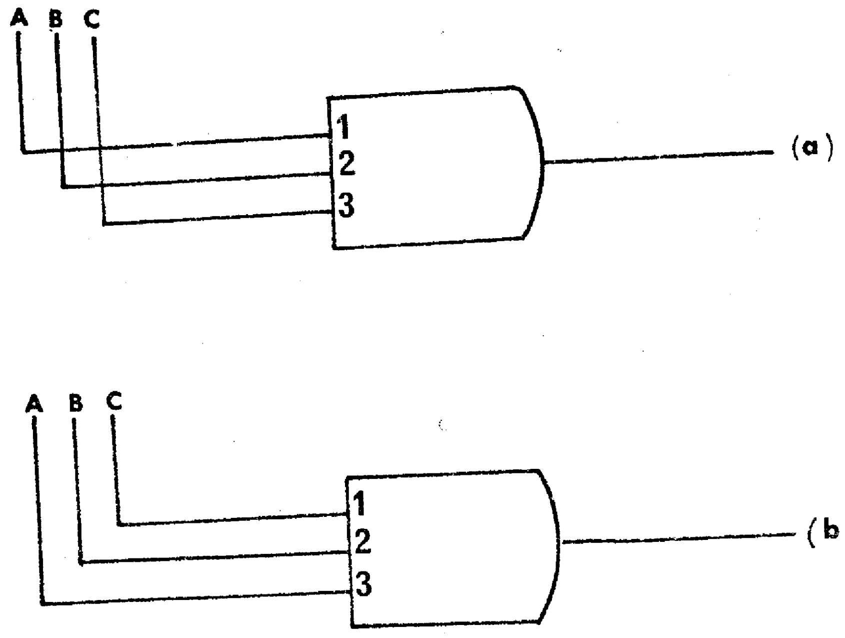


Fig. 1.1.1 Influence of pin assignment on the number of crossings in a logic diagram.

possible layouts involving a 3-input AND gate are shown. It is obvious that the layout of Fig. 1.1.1(b) is much better than the one in Fig. 1.1.1(a), while logically there is no difference. A good layout algorithm will have to detect such properties and use them accordingly. This problem is sometimes referred to as the pin assignment problem and is even more important for the layout of printed circuits. Connections are allowed to cross each other as long as these crossings do not influence the clarity (readability) of the logic diagram. As mentioned before, clarity is a very subjective measure of quality.

Printed circuit layout has received a lot of attention from the industry because algorithms and methods have been developed that allow automated printed circuit layout, that is usually more economical than manual layout. Most of these algorithms are particularly suited for the layout of multi-layer boards with regularly shaped components (i.e. integrated circuit modules). Much less work has been done on the layout problem of one-sided boards with a large variety of components (mainly discrete components). One common characteristic that differentiates the printed circuit layout problem from the logic schematic layout problem is that two distinct connections are not allowed to cross. In the case of one-sided boards, components are normally placed on one side, while the connections are realized on the other side of the board. On two-sided boards connections can be made on both sides. Depending on the technology used, a connection must lie completely on one side of the board or can consist of segments on both sides, connected by plated-through holes or vias. Vias can be fixed or floating. Fixed vias can only be placed in predefined locations, while floating vias can be located where they are needed. In most cases, the aim of the design is to minimize the total wirelength and/or the number of vias. Other parameters include the length of the longest wire, the size of the board, etc. It is obvious that any connection problem can be solved using a two-sided board with vertical segments on one side and horizontal segments on the other side, provided the board is large enough. This may result in an excessive number of vias. Multilayer boards (4,6, ...16 layers of interconnections) allow a much smaller board area, result in much shorter interconnections, but are more expensive to manufacture. The circuit layout problem here is to assign interconnections to the appropriate layer and to place the modules in such a way that all interconnections can be realized.

The layout of integrated circuits is related to printed circuit layout, with some important differences. A number of components of different shapes and sizes are to be connected using one or two layers of interconnections. In addition to these layers, one can use "cross-overs" that allow a connector on these layers to cross one or more connections on the same layer. A cross-over however consumes some physical area of the integrated circuit chip and should be avoided when possible. The parameter to be optimized here is the total area of the chip.

1.2 Methods for Automated Circuit Layout.

1.2.1. Introduction.

Most procedures for solving the circuit layout problem first place the components on the board, thereby minimizing an objective function. This function should be a measure of the quality of the final layout. Usually the total wirelength is the parameter one tries to minimize. This tends to cluster heavily connected components together and to shorten the longest wires, which are two desirable side-effects. Once the placement is obtained, it is frozen and the routing of connections has to be performed within this fixed-component topology. Pin and gate assignment are often done in a post-processing phase.

Topological layout methods, on the other hand, 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 then is embedded in one or more planes with the restriction that no two edges must intersect except at the vertices. If some of the connections remain unembedded, one attempts to route them by making use of technological properties. The final step consists in transforming the topological layout into a physical layout, that takes into account the geometrical properties.

Many existing systems for topological IC layout e.g. Engl (EM73), Klamet (Kl73), Sugiyama (Su74) are limited to small-scale circuits. Because of the inadequacy of the models and algorithms employed, they often rely heavily on interaction for obtaining a final layout.

1.2.2. Geometrical versus Topological Information.

The topological aspect of the circuit layout problem relates primarily to the manner in which components are interconnected. Further topological information includes the order in which the terminals of a component appear on its outer boundary as well as the possibility of routing connections under or over the area used by the component. The requirement that the external connections have to appear on the outside boundary of the circuit in a prespecified order is also a topological characteristic of the circuit layout problem. Sometimes, the order of terminals is not completely imposed upon the designer: e.g. the inputs of a three-input AND gate are interchangeable and a good layout procedure should take this into account.

The geometrical aspect of the circuit layout problem is primarily related to parameters that can be measured. For layout problems one does not use the ordinary Euclidian metric, but rather the so-called Manhattan geometry, in which only vertical and horizontal line segments are allowed. The size of individual components, the thickness of conductor lines and the size of a printed circuit board or an integrated circuit chip are examples of geometrical parameters.

The classical approach, where placement and routing are performed independently, takes into account the geometrical information at all stages, as long as there is a predefined regular structure. The topological parameters, however, are not fully taken into account.

In the topological approach, the topological parameters are considered at all stages, while the geometrical information is used only in the last phase of the layout.

The classical approach has proven succesful in the layout of multilayer printed circuit boards with a regular structure, where the number of crossings is not important. This method has also been applied to the design of large scale integrated circuits using the block-and-track method (Koz72). This method for LSI layout is used extensively today, but it results in a rather inefficient use of the silicon area available. Basically, one has a library of predesigned components (e.g. NAND, NOR gates, flip flops etc.), where each basic cell has the same height and a different width. The cells are placed in rows and connections can be routed in predefined channels. By imposing these restrictions, one can apply the classical techniques for placement and routing. Although this method allows rapid layout of LSI circuits, it does not make good use of the silicon area. Currently, most circuits for high-volume production are designed manually in order to obtain an optimal packing density. This manual process can take a few months.

1.2.3 The Classical Approach.

Most methods for solving the circuit layout problem first place the elements such that some objective function is optimized (in most cases the total wirelength). Then the interconnections are routed in a separate step.

The placement problem for printed circuits can be formulated in the following way:

The board is divided into a number of rectangles of equal size, called slots. Given n modules $M(i)$, $i=1,n$ and m slots $S(j)$, $j=1,m$ where $m \geq n$, assign to every module a distinct slot such that a specified function is minimized. Every module has a number of pins that are to be connected to other pins in order to implement a design. On the set of all pins one can define a relation "is connected to". This is an equivalence relation that partitions the set of all pins into a collection of disjoint subsets. These subsets are usually referred to as the <u>nets</u>. The objective function that one usually attempts to minimize is the total wirelength. Minimum total wirelength is only one of many real-life parameters one might try to minimize. Heat dissipation, signal cross-talk, routeability of the connections, size of the board, etc. are to be taken into account. It has been determined empirically that, by minimizing the total wirelength, one usually satisfies most of the other requirements reasonably well.

Assuming all nets to be of cardinality 2, the total wirelength can be written as:

$$\underset{i,j}{\text{SUM}} \quad c(i,j) * d[p(i),p(j)]$$

where c(i,j) is the number of wires between modules i and j and d(k,l) is the distance between slots k and l; d[p(i),p(j)] is the distance between the slots to which modules i and j have been assigned. The function has to be minimized over all possible permutations.

This problem is known as the quadratic assignment problem, which is in itself a special case of the more general module placement problem, where nets can be of cardinality > 2. It has been shown by Garey et al. (GJ74) that the quadratic assignment problem is NP-complete. Therefore, it is very unlikely that it can be solved in polynomial time. Hence, approximations are frequently used. Steinberg (St61) proposed an iterative method where in each step a linear assignment problem on a set of mutually unconnected components is solved. Hanan and Kurtzberg (HK72a-b) use an 'initial placement' technique to find a first placement of the components. This initial placement is then improved by means of a 'placement improvement' method such as Steinberg's. Another placement method not requiring slots, is the 'force-vector' method originally developed by Fisk,Caskey and West (FC67a-b) and further described by Hanan and Kurtzberg (HK70a). Module placement algorithms for problems with modules of widely different sizes (integrated circuits, printed circuit boards with discrete components) have received much less attention.

Solutions to the routing problem can be classified in two major categories:

1) maze-running algorithms, that require the board to be divided by a grid such as Lee-type algorithms that are described further in (Le61), (Ma64), (LM64) and (Ak67). These algorithms route one connection at a time and require time proportional to N**2 per connection, where N is the length of the path; an advantage is that this type of algorithm always finds a path when one exists. Furthermore it is very flexible in the parameters to be minimized (e.g. wirelength, number of vias, etc.). A depth-first approach, as proposed by Rubin (Ru72), yields more efficient algorithms.

2) Another class consists of the channel routing algorithms, where the available space is divided into channels (e.g. Stevens (St72), Stevens and Hashimoto (HS71)). These algorithms require much less storage space and are considerably faster. Some route one wire at a time, while others (Stevens) route all wires in parallel. A disadvantage of channel routing is that it requires a regular board structure such as slots and channels, while maze-running algorithms do not have such requirements.

Another routing algorithm, quite different from the previous ones is Hightower's line-search algorithm (Hi69). In this algorithm, connections are routed one at a time. Since no grid is stored (as in Lee-type algorithms), its storage requirements and run time are much less while giving results comparable to Lee-type algorithms. Hightower's algorithm does not require a regular board structure, which makes it more generally applicable than channel routing algorithms.

A more detailed survey of interconnection routing is contained in (Ak72) and (Na72).

Although many good algorithms have been developed for solving both the placement and the routing problem, very few attempts were made to solve both problems simultaneously.

1.2.4 The Topological Approach.

Topological methods use a very different approach: they are based on graph theory concepts, mainly planarity.

The main concern in solving the circuit layout problem is to embed the connections in one or more planes, such that no two connections intersect. This criterion shows a striking similarity with the planarity concept in graph theory: a graph is planar if it can be embedded in the plane such that no two edges intersect.

Although several attempts were made to solve the circuit layout problem using planarity related methods, working systems have appeared only recently.

Topological methods for laying out one-sided printed circuits were proposed by Kodres (Ko62) and Weissman (We62). Methods for the layout of thin film RC circuits were mentioned by Sinden (Si66ab) and Bedrosian (Be67). This problem however is very restrictive: there is only one layer of conductors while crossings can occur only at the capacitors. Weinberg(We68) discusses graph-theoretical concepts such as planarity and isomorphism, that are useful for solving layout problems. Akers and Hadlock (AH69) describe a layout method for IC's based on a graph-theoretical method. Akers, Geyer and Roberts (AG70) continue this approach and also describe a method to transform the planarized graph into a physical layout, which takes into account the actual dimensions of the components.

A good survey of the topological approach to the circuit layout problem is given by Kodres (Ko69). Working systems for the layout of integrated circuits, based on a graph-theoretical approach, are described by Yoshida and Nakagawa (YN69), Engl and Mlynski (EM69a-b,EM73) and Fletcher (Fl72). An effort to justify theoretically the models used is given by Engl and Mlynski (EM72abc,EM75).

Ulrich (Ul68, Ul69, Ul70) describes a model to take into account connector orientation, but no information is available on a practical implementation.

One serious objection to topological layout methods is that they usually do not take into account any physical parameters, such as the number of wires one can route between two pins or the capacity of a routing channel. As will be indicated in a later chapter, it is possible to take finite capacities into account in a graph theoretical model.

One should ask the question whether it makes sense to have channels and slots? Routing channels were introduced partly to simplify the problem algorithmically. The same is true for slots: they simplify the placement problem. One disadvantage of slots and channels is that this restriction can lead to bad utilization of the available space. On the other hand, it easier to manufacture printed circuit boards with a regular structure. Integrated circuits however do not have these manufacturing characteristics; they are frequently composed of components of very different size and shape. Furthermore, it is important to minimize the total IC chip area (at least for large scale fabrication).

1.3 Some Existing Graph Models for the Circuit Layout Problem and their Properties.

In this section, a number of existing graph-theoretical models for the circuit layout problem will be discussed. In each case, the advantages (if any) and the disadvantages of these models will be explored.

A proper model for the circuit layout layout problem should take into account the following properties of physical circuits:

1) The model should be planar if and only if the physical circuit is planar.

2) The order of the terminals of a component can be completely specified or it can be partly or completely unspecified. For example, the nets connected to the inputs of a 3-input AND gate can be interchanged without changing the logic function of the circuit.

3) Sometimes a mirror-image component is available (e.g. in integrated circuit layout).

4) In some cases, connections may be routed under or over a component's physical area, while in other cases, this is not allowed.

5) the model should avoid the decomposition a priori of a multi-terminal net into a number of simple connections. If possible, this decomposition should be done in function of the layout.

1.3.1. Module-to-vertex, Connection-to-edge Mapping.

It is not possible to represent a net by a single edge since an edge expresses a binary relationship (adjacency) between two vertices of a graph. A simple interconnection on the other hand represents the binary relationship of two modules being interconnected. Therefore, when building the graph model of a circuit, one has to decide a priori how a net will be decomposed into simple connections. This can be formulated as finding a spanning tree in a complete graph on N vertices, where N is the number of points to be connected by the net. For a net, connecting N points, there are N**(N-2) possible decompositions. This net decomposition can introduce non-planarities in the model as

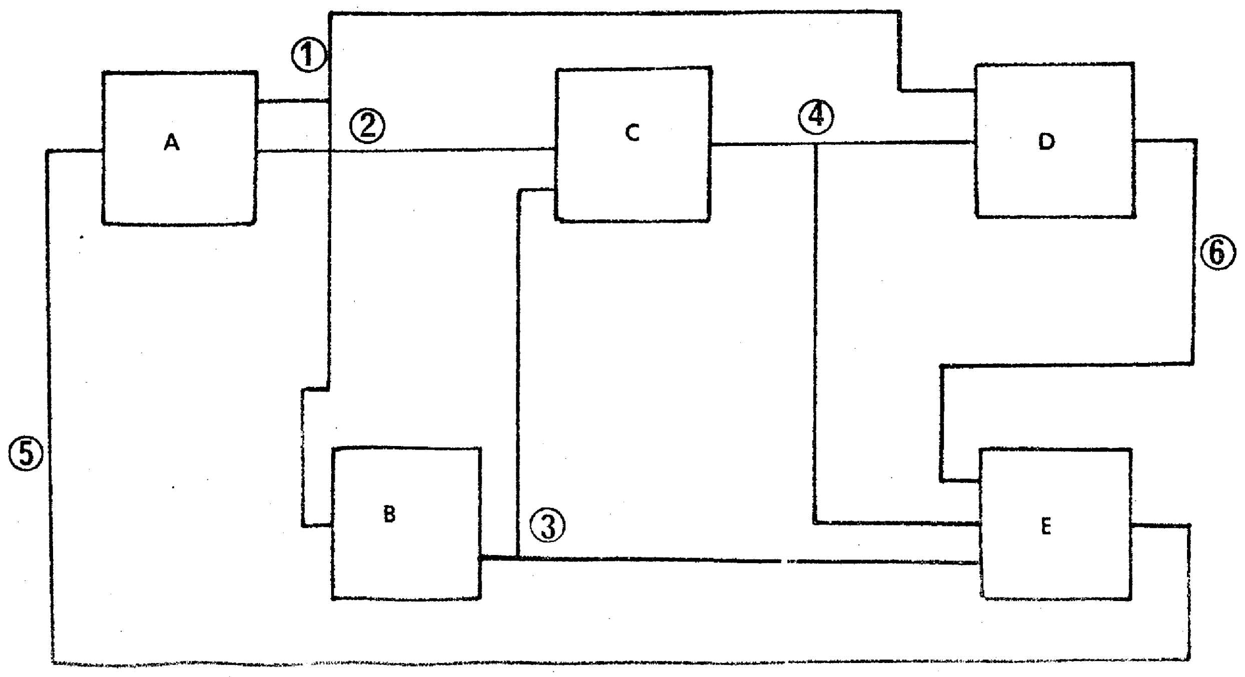


Fig. 1.3.1 Example circuit 1.

will be shown later. First consider the example given in Fig. 1.3.1 : the circuit shown has 5 modules and 6 nets. Assume the following arbitrary decomposition of the nets:

(A,B,D) into (A,B) and (A,D)

(B,C,E) into (B,C) and (B,E)

(C,D,E) into (C,D) and (C,E). For this decomposition, it is possible to construct a graph; this is done in Fig. 1.3.2 . This planar graph could have been obtained by using an algorithm for embedding the graph in the plane. Note that edges 1' and 1" are separated by edges 2 and 5. Edges 1' and 1" are physically part of the same net and are

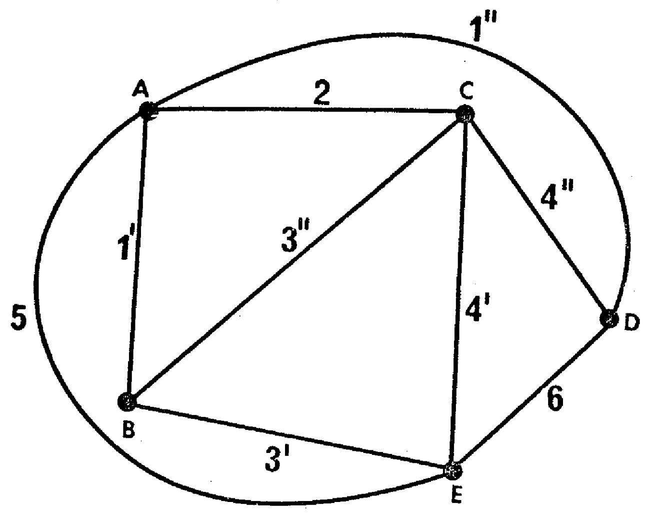


Fig. 1.3.2 Module-to-vertex, connection-to-edge mapping for example 1.

interconnected to the same pin of module A.

Although the graph used as a model is planar, the physical circuit is not. The circuit could be made planar if connections were allowed underneath the modules. This is sometimes possible: e.g. for certain types of modules on a printed circuit board. In general it is not possible for discrete components (such as transistors) or for the 'black boxes' on a logic diagram.

Consider the second example of Fig. 1.3.3: this is a circuit with 6 modules and 6 nets.
As can be seen from this diagram, the circuit is planar. The nets may be decomposed as follows:

(A,D,E,F) into (A,D), (A,E) and (A,F)

(D,B,C) into (D,B) and (D,C)

(E,B,C,F) into (E,B), (E,C) and (B,F).

The result of this decomposition is shown in Fig. 1.3.4. This graph is non-planar since it contains K(3,3) as a subgraph. †

---

† The notation K(n) and K(m,n) will be used to indicate the complete graph on n vertices and the complete bipartite graph on n and m vertices respectively. Consequently, the Kuratowski subgraphs will be referred to as K(5) and K(3,3).

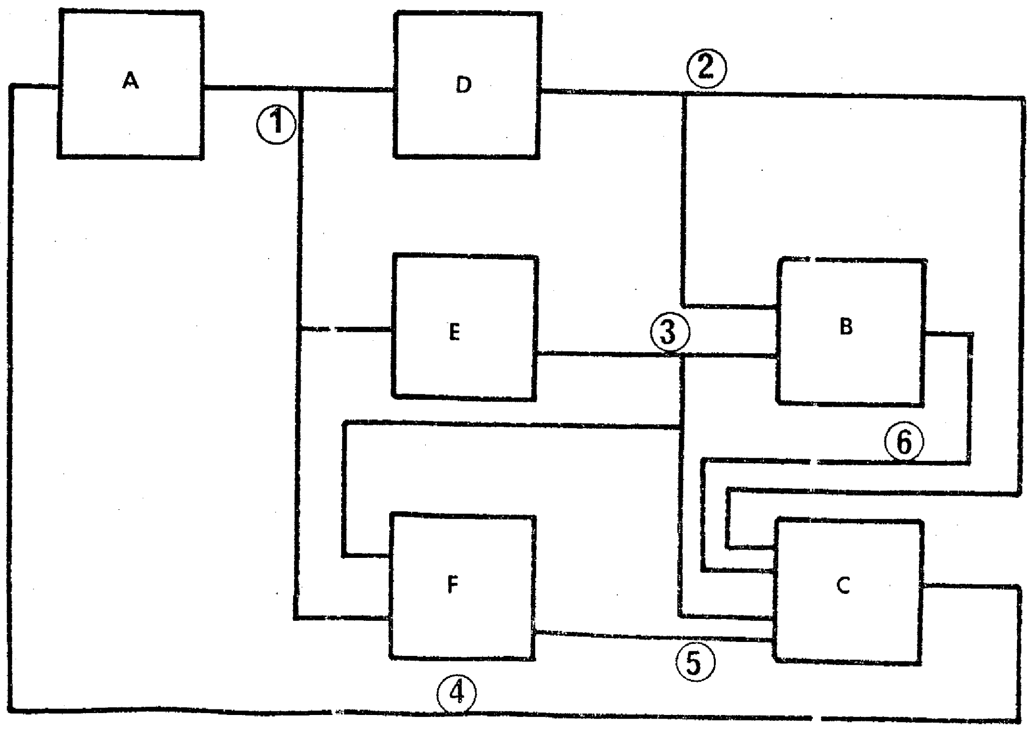


Fig. 1.3.3 Example circuit 2.

We can conclude that mapping modules into vertices and connections into edges is not a proper model for the circuit layout problem since:

1. The graph model can be planar while the circuit is not (first example).

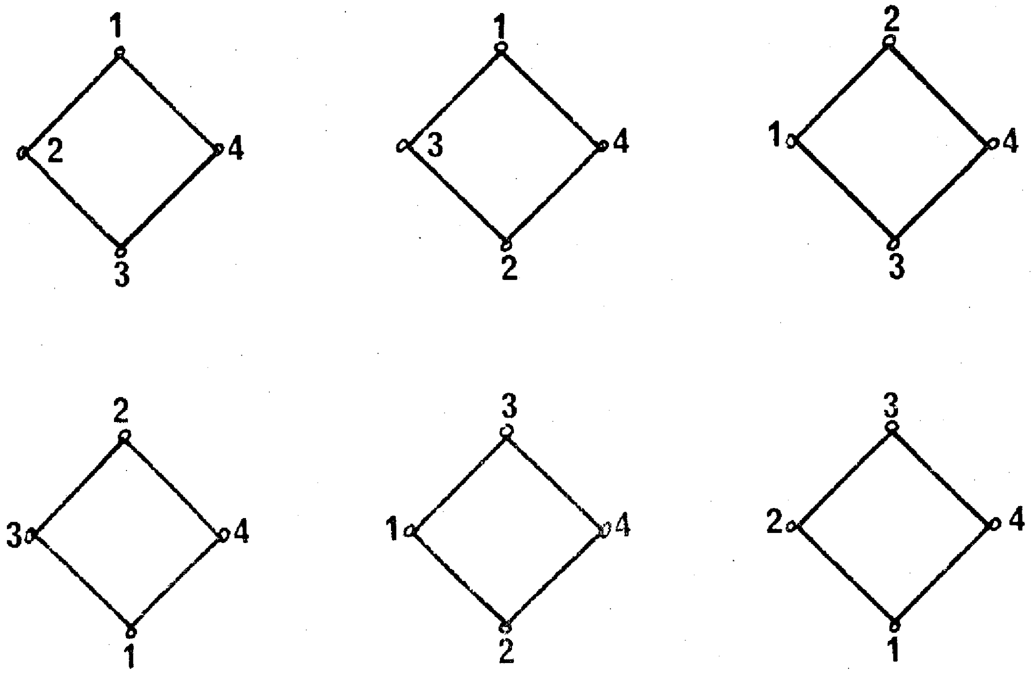


Fig. 1.3.4 Module-to-vertex, connection-to-edge mapping for example 2.

2. The graph model can be non-planar when the circuit is planar (second example).

3. The model does not take into account whether connections can be located underneath modules or not.

4. One is required to guess a priori how a net should be decomposed into simple connections.

This model was used by Yoshida and Nakagawa (YN69).

1.3.2. Module-to-cycle, Net-to-vertex Mapping.

Here the pins of a module are represented by vertices connected to each other by edges forming a cycle for modules with 3 or more pins. Two-terminal modules are represented by single edges. Representing a module with n pins by a cycle can be done in n! different ways, one of which has to chosen: this problem is somewhat similar to chosing a spanning tree to represent a net in the first model. Fortunately, in many cases, the order of the terminals of a module is important and specified. This makes the choice obvious. Sometimes, the order of the terminals is only partly specified; e.g. consider a 3-input AND gate with input pins 1, 2 and 3 and output on pin 4. As shown in Fig. 1.3.5, there are only 6 different possible models instead of 16 because the three inputs always should appear in a 'cluster'.
We therefore have to evaluate this model for 2 distinct cases:

1. The order of the terminals is partly or completely unspecified. Then we have to choose one out of many possible cycles to represent the module. Consider the

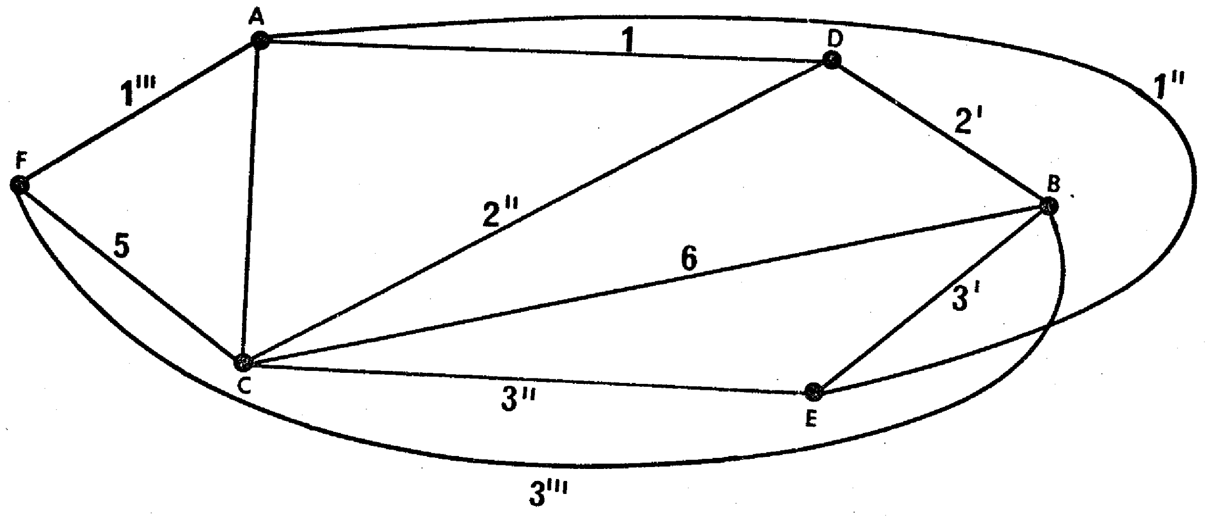
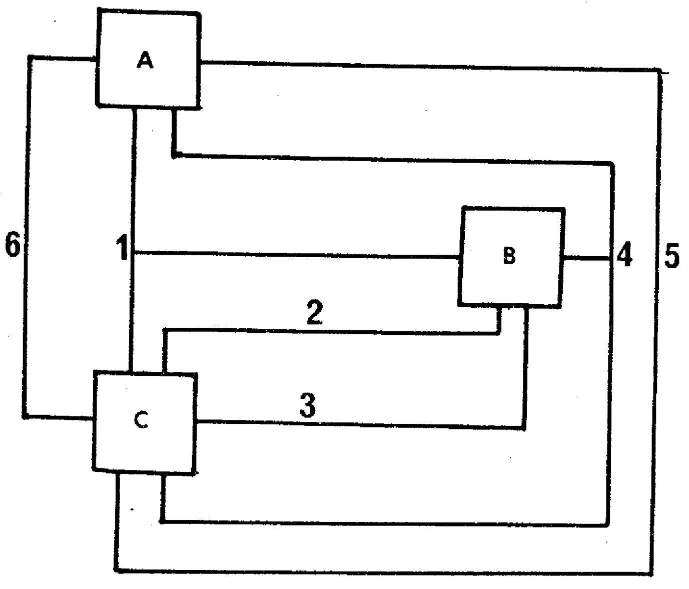


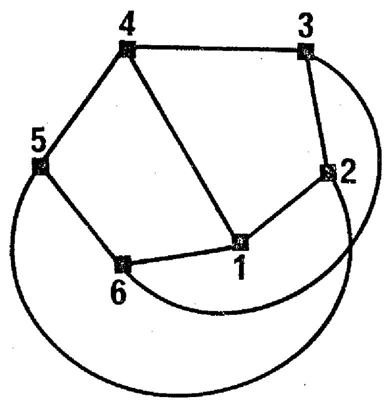
Fig. 1.3.5 Possible models for a 3-input AND gate, assuming a module-to-cycle, net-to-vertex mapping.

example given in Fig. 1.3.6, where three modules A, B and C are connected by 6 nets.

Assume that for module C, the terminals, connected to nets 3, 4, 5 and 6, are interchangeable. Let module A be represented by cycle (1,6,5,4,1) and



(a)



(b)

Fig. 1.3.6 (a) a planar circuit. (b) its non-planar model, using a module-to-cycle net-to-vertex mapping.

B by cycle (1,4,3,2,1). If we select the cycle (1,2,3,4,5,6,1) to represent module C, then the graph model is planar. However, if we select (1,2,5,6,3,4,1) as the cycle to represent the module, then the resulting graph model is non-planar (Fig. 1.3.6 (b)).

2. The order of the terminals of a module is specified. Here the use of a cycle to represent modules is an improvement over the use of a single vertex. Several papers have proposed this as a model for modules, e.g. Rose (Ro70), Basden and Nichols (BN73), Ulrich (Ul68,Ul70). However, representing the net by a vertex that coincides with a vertex representing a pin of a module creates problems: the circuit of Fig. 1.3.7(a) is non-planar, while its model (Fig. 1.3.7(b)) is planar.

A modified version of this model was proposed by VanLier and Otten (VO73). They represent a module by a wheel-like subgraph: one special vertex for the module itself and one vertex for each pin. The vertices representing pins coincide with the nets. The special vertex for the module is connected by edges to all vertices representing the pins of that module. The pin vertices are also connected by edges forming a cycle, modelling the circular relationship between the pins of the module. A result of this model is that no connections are allowed underneath the module. Another problem that can occur when using a cycle as a model for a component is that an

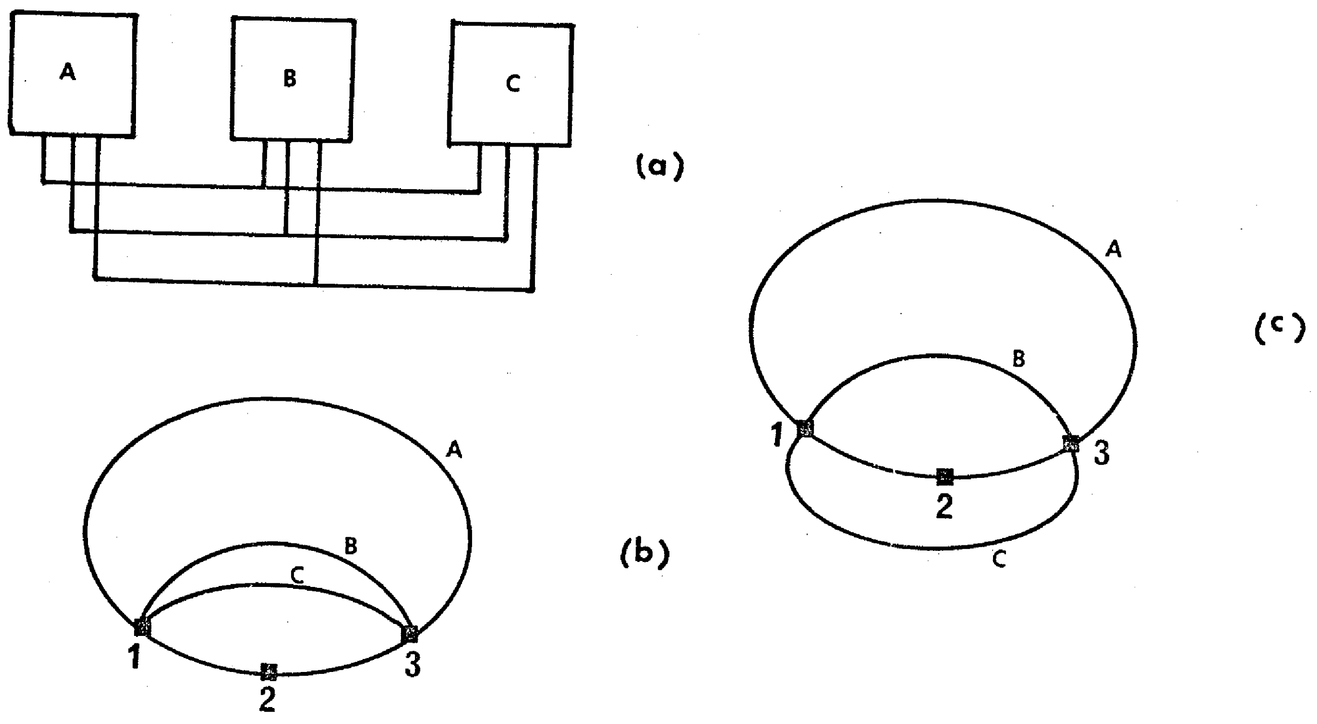
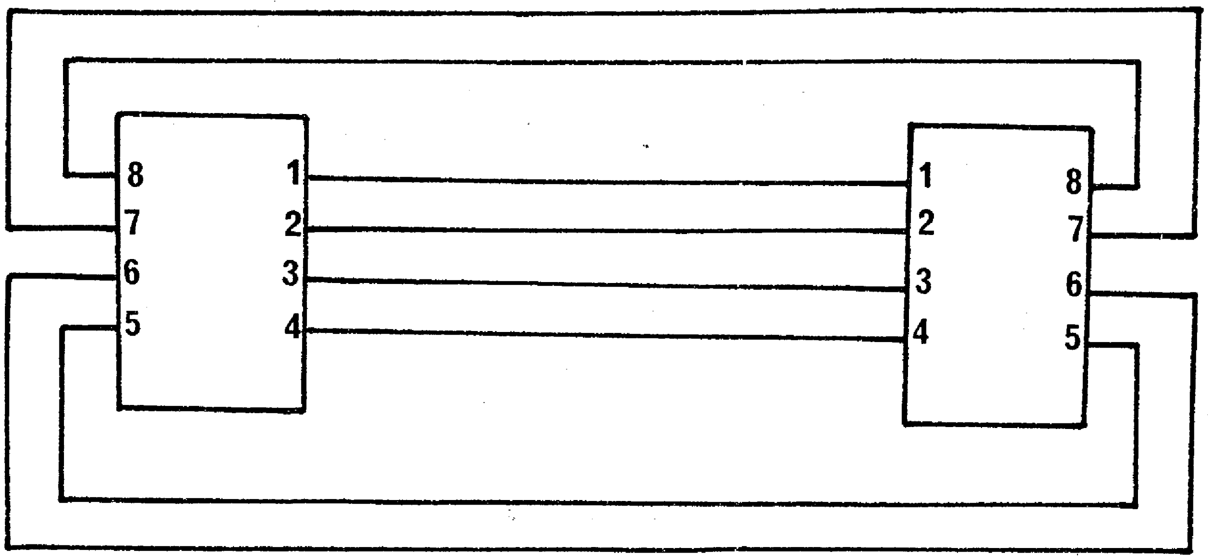


Fig. 1.3.7 (a) a non-planar circuit. (b,c) Planar models, assuming a module-to-cycle, net-to-vertex mapping.

embedding might result in a plane graph in which the mirror image of the cycle occurs. This is once again dependent on the technology used: for printed circuits it means that the component has to be placed on the other side of the board, which is impractical. In IC technology a symmetric component is usually available. An example of such a situation is given in Fig. 1.3.8. Van Lier and Otten (VO73) consider this problem and propose a 'post-mortem' method, where the graph representing the circuit is first embedded in the plane and then is checked for correct orientation of the multi-terminal components. In order to do this, the graph is broken up into biconnected components. If the orientation of all critical cycles in one of the biconnected components is not the same then the method fails. Otherwise, if the orientation of all critical cycles in one component is different from the orientation in another component, then one of these components is rotated around its points of articulation. This approach is clearly unsatisfactory in cases such as the one illustrated by Fig. 1.3.8. This planar biconnected graph cannot be corrected by this method.

A better method is to use a constructive planarity testing and embedding algorithm, that takes into account the pre-specified orientation of certain cycles. Such an algorithm will be described in chapter 4.



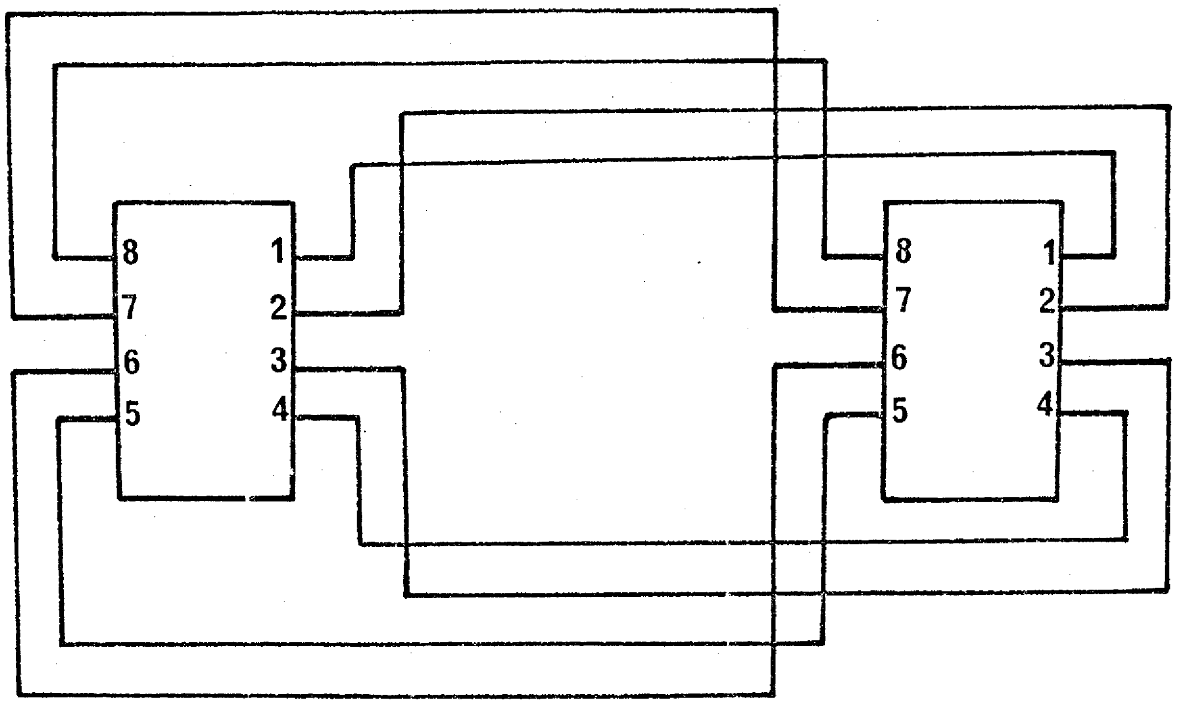


Fig. 1.3.8 Influence of the component orientation on the planarity of a circuit.

1.3.3. Module-to-vertex, Net-to-star Mapping.

This is the model proposed by Goldstein and Schweikert (GS73). Each component is represented by a vertex while a net connecting k components is represented by a vertex and k edges connecting this vertex to the vertices representing these components: a net is modelled by a 'star'-like subgraph. The model for the circuits of Fig. 1.3.1 and

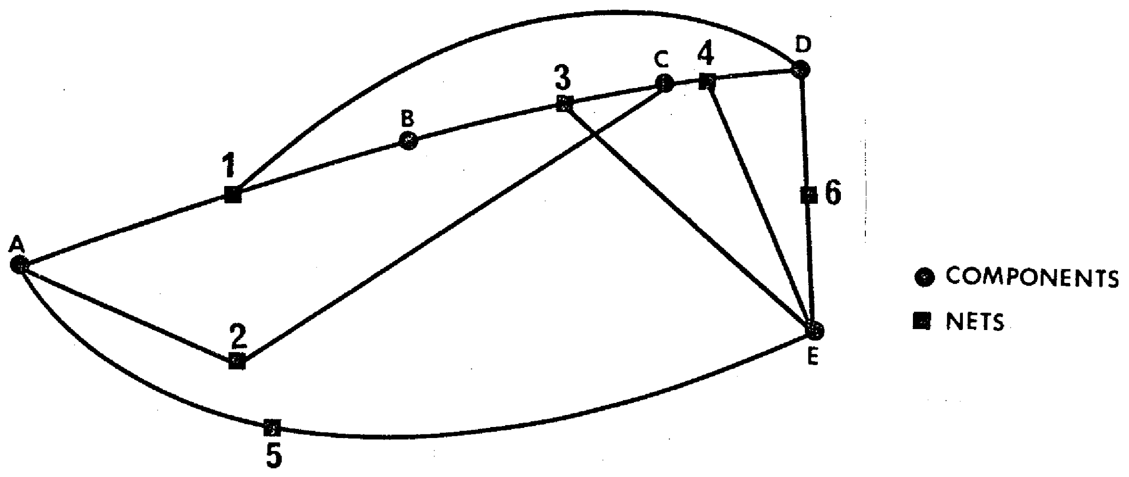


Fig. 1.3.9 Module-to-vertex, net-to-star mapping for example 1.

1.3.4 is shown in Fig. 1.3.9 and 1.3.10.

If the order of the terminals is irrelevant, then this model is usually adequate for the purpose of testing the circuit for planarity. The following disadvantages should be noted:

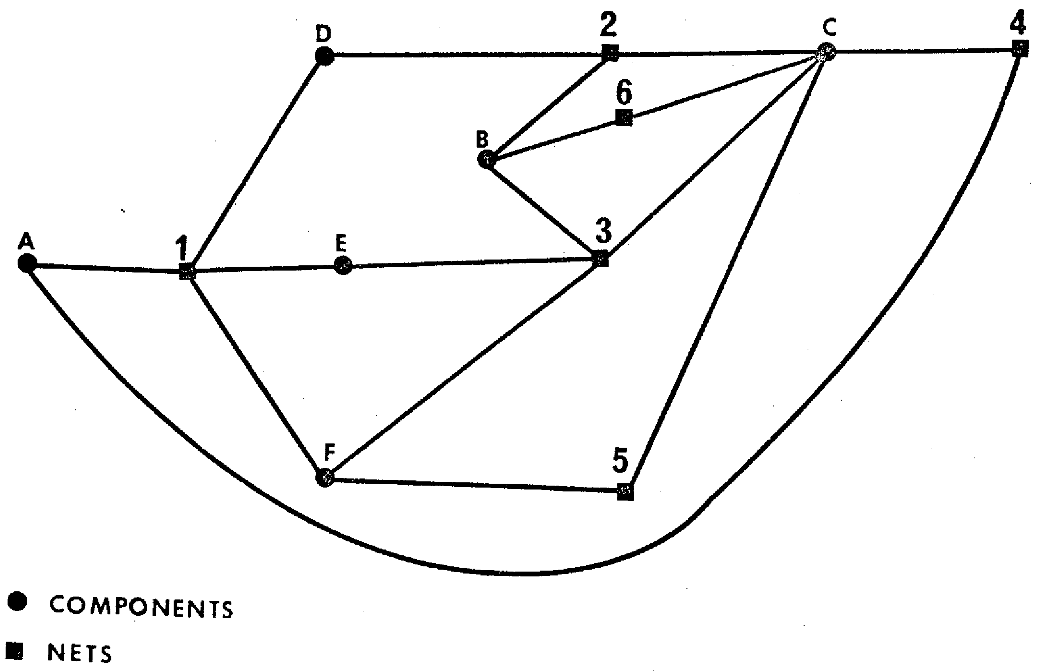
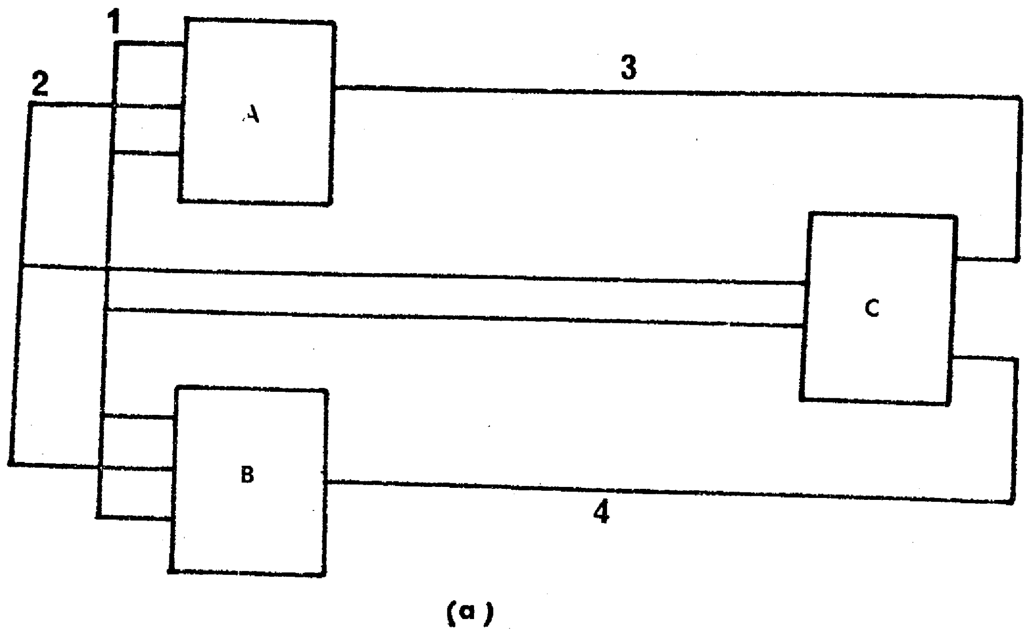


Fig. 1.3.10 Module-to-vertex, net-to-star mapping for example 2.

1. If the technology restricts the cyclic order of the terminals of a module, then the model will not show crossings that in reality are necessary. An example of such a case is given in Fig. 1.3.11.

2. The model does not take into account the fact that it is often possible to route connections under modules or between pins of a module.



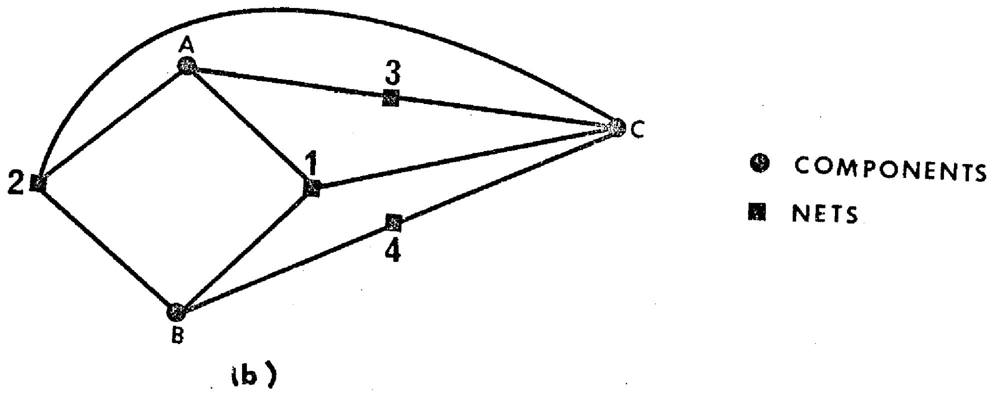


Fig. 1.3.11 Module-to-vertex, net-to-star mapping for a circuit where the pins have a pre-assigned order.

3. The graph model can be non-planar for a planar circuit when a net can be connected to one of several possible terminals of a component. An example of such a component is an integrated circuit transistor as shown on Fig. 1.3.12.
   It can be seen easily that this model is not capable of correctly representing the situation shown in Fig. 1.3.12.b and c, where three IC transistors are connected in parallel: the real circuit is planar while the model is non-planar.

It should be mentioned here that the resulting graph is bipartite with the vertices representing the components being one partition and the center vertices of the stars representing the nets being the other partition.

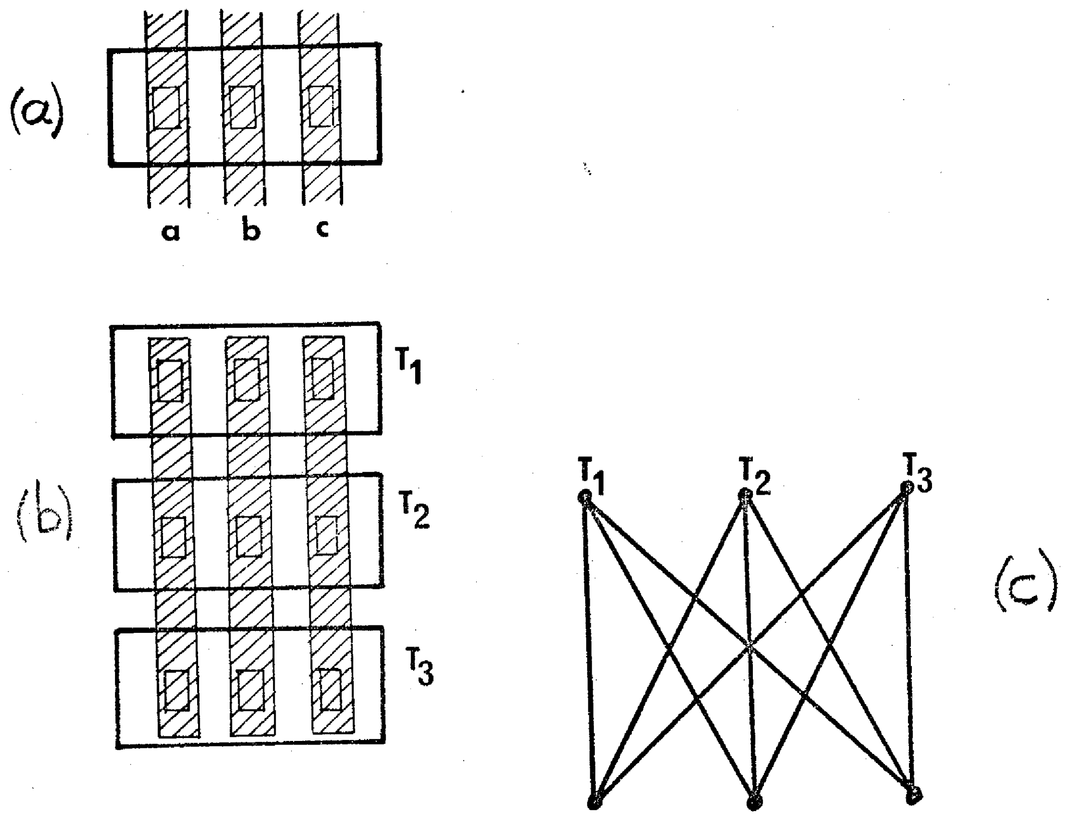


Fig. 1.3.12 (a) IC Transistor.
(b) Three IC Transistors in parallel.
(c) Module-to-vertex, net-to-star mapping for (b).

1.3.4. Module-to-star, Net-to-vertex Mapping.

Here a n-terminal component is represented by a vertex and n edges connecting this vertex to the vertices representing the nets to which this component is connected. A model very similar to this was proposed by Engl and Mlynski (EM69ab, EM72abcd, EM73). The difference is that they represent a 2-terminal component simply by an edge rather than by a vertex of valency 2 and 2 edges. For the purpose of testing the planarity, there is no difference, since the two graphs are homeomorphic.

The model for the circuits of Fig. 1.3.1 and 1.3.3 is shown in Fig. 1.3.13 and 1.3.14. This mapping results in a bipartite graph while Engl and Mlynski's model does not.

It should be noted here that this model is identical to the previous one proposed by Goldstein and Schweikert (GS73) and as such it has the same disadvantages. A common formulation for both models is the following: Components are represented by a set A of vertices; nets by a set B of vertices. Whenever a component is connected to a net, there is an edge connecting the corresponding vertices. The

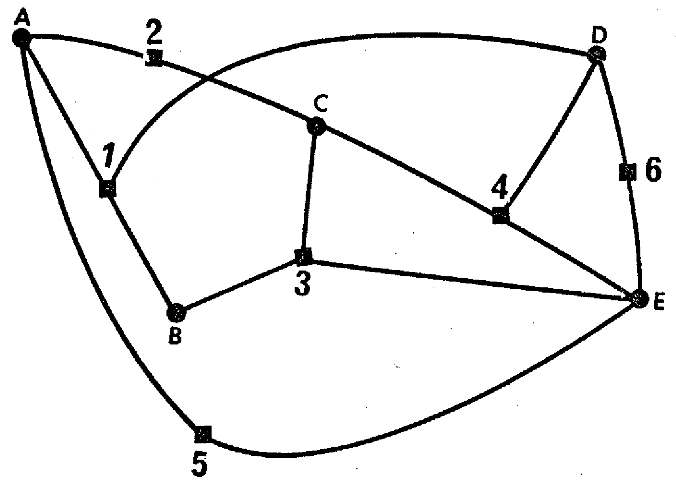


Fig. 1.3.13 Module-to-star, net-to-vertex mapping for example 1.

vertex set consists of two partitions A and B and the graph is clearly bipartite.

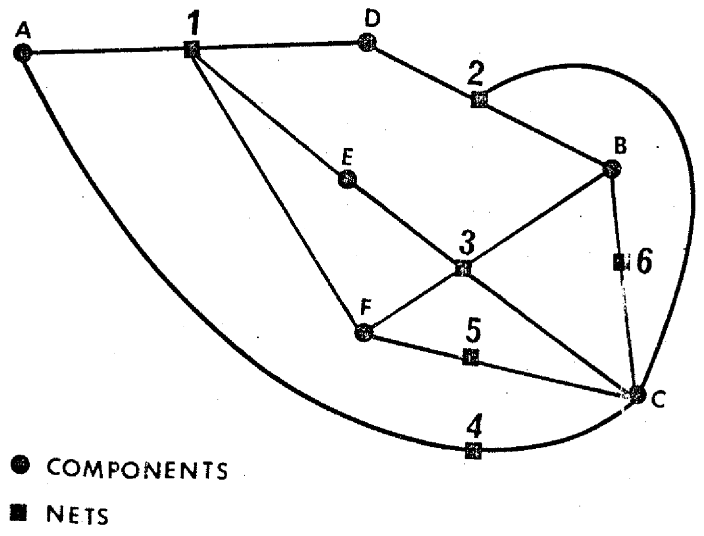


Fig. 1.3.14 Module-to-star, net-to-vertex mapping for example 2.

1.3.5. Module-to-cycle, Net-to-star Mapping.

In this model, pins are mapped into vertices. In order to represent the cyclical order of the pins of a given module, edges are used to model this relationship. A net is represented by a vertex that is connected by edges to all pins that belong to that net. A model of this kind was proposed by Rose (Ro70).

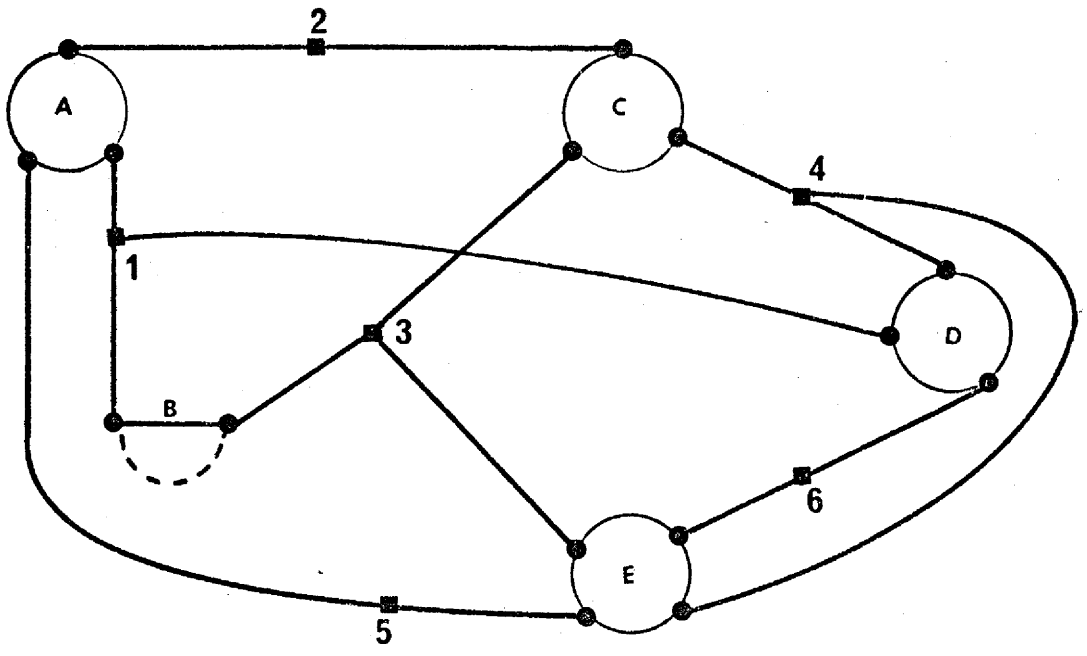


Fig. 1.3.15 Module-to-cycle, net-to-star mapping for example 1.

Fig. 1.3.15 and 1.3.16 show this model for the circuits of Fig. 1.3.1 and 1.3.3. This model combines the good features of the previous models, such as correct representation of the nets and of the cyclical order of the pins. Connections are allowed underneath the modules. In order to prevent this, one could add a vertex inside each cycle and connect it to all pins of that module as was mentioned by VanLier and Otten (VO73).

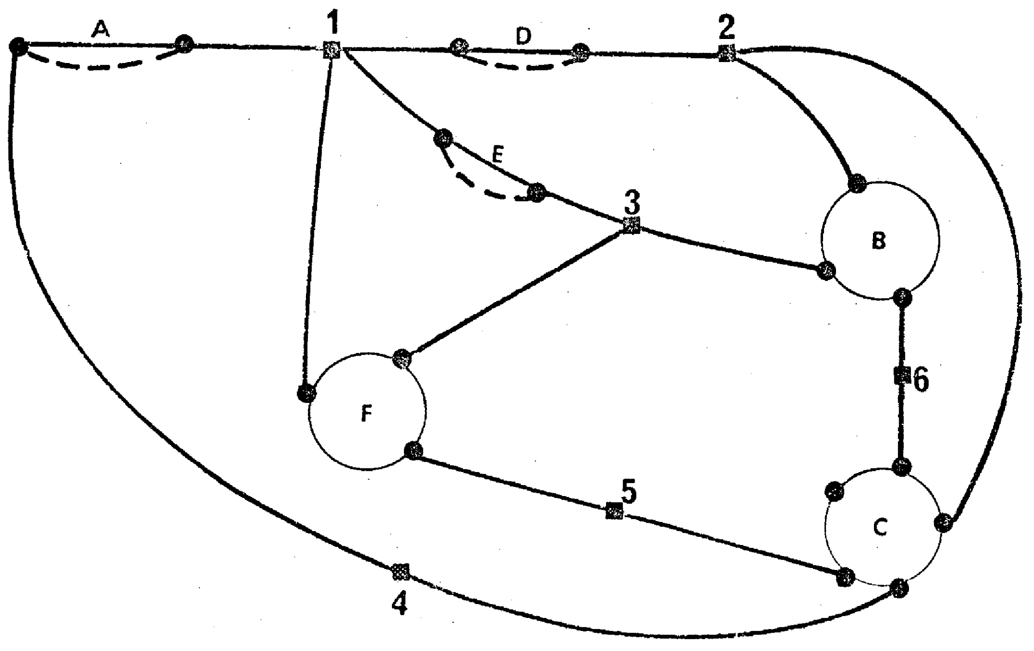


Fig. 1.3.16 Module-to-cycle, net-to-star mapping for example 2.

1.4 On the Representation of Boundary Conditions.

Printed circuit boards have one or more external connectors that normally lie on the periphery of the board. In IC technology, a number of bonding pads has to lie on the chip's periphery. In many cases the order in which the terminals occur is partly or completely specified. In most cases (e.g. Rose (Ro70), Weinberg (We68), Basden and Nichols (BN73)) the terminals are represented by vertices connected by edges in the prespecified order (cycle). Engl and Mlynski (EM69ab) and VanLier and Otten (VO73) represent the connector itself by a star-like subgraph with the center vertex for the connector itself and a vertex for every pin. In addition to that they model the sequence of the terminals by a cycle. The purpose of any model is to obtain a graph, that when embedded, has the terminals on the periphery and in the correct order. The only feasible way of doing this is to use an embedding algorithm that forces the peripheral cycle to be a face.

None of these models takes into account that the order of the terminals might only be partially specified. e.g. when designing an IC containing a 4-input AND gate, these 4 inputs are completely interchangeable as far as their sequence on the periphery is concerned.

1.5 Conclusions.

None of the models described takes into account the following factors:

- some pins are interchangeable; e.g. the inputs of an AND gate; normally pins are assigned to nets before the actual layout is done (pin assignment).
- some parts of a module are interchangeable; e.g. 4 identical gates in a dual-in-line package; again these gates are assigned before the layout is done (gate assignment).
- sometimes wires are allowed between two pins of a module or underneath it, but the number of such wires is limited. Cross-overs in IC technology have a certain size, allowing only a few wires to use it. In short, finite wiring capacities are not taken into account.

- the order in which outside connections have to occur is frequently only partially specified. Current models require the sequence to be completely specified and as such force a decision to be made that might adversely affect the layout of the circuit.

2 An Improved Graph Model for the Circuit Layout Problem.

2.1 Introduction.

From the previous chapter, it is clear that a model for the circuit problem has to take into account the following

- whether or not the order of the terminals of a component is prespecified.

- if the order of the terminals is given, some of these terminals can be interchanged without changing the logic of the circuit (e.g. the inputs of an AND gate). This property will be referred to as the _logical equivalence_ of terminals. The pin assignment problem consists of assigning nets to logically equivalent terminals as a function of an optimal layout.

- sometimes a net can be connected to one of several possible terminals of a component (e.g. an IC transistor, as shown in Fig. 1.3.12). This property will be referred to as the _physical_ _equivalence_ of terminals.

- some parts (subcircuits) of a component can be interchanged (e.g. several identical AND gates in a dual-in-line package); The gate assignment problem consists of assigning groups of nets to logically equivalent subcircuits as a function of an optimal layout.

- some components allow wiring underneath, while others do not.

- in some cases, it is possible to route one or more connections between two adjacent terminals of a component, while in other cases it is not. Finite wiring capacities of this kind should be represented in the model.

In this chapter, we will derive an improved model, that takes into account most of these requirements. The theoretical basis for this model will be given in the next chapter.

2.2 A Model for Nets.

It will be assumed here that pins are represented by vertices and further abstraction will be made of the model used for the components. Basically, a net connecting n pins can be realized in many ways: any connected spanning subgraph of the complete graph on n vertices will do. Since loops are redundant, we can restrict ourselves here to all spanning trees of the complete graph on n vertices, of which there are n**(n-2). For all nets, every possible combination has to be considered in order to find the minimum number of crossings. This is a combinatorial problem, for which the only solution method is to enumerate all possible combinations. This is clearly impractical. When considering the model proposed by Goldstein and Schweikert (GS73)(i.e. a star-like subgraph), one may ask whether this is a good approximation. It can be proven that if a graph model with stars representing the nets is planar, then there exists a spanning tree for the nets such that the graph is planar. On the other hand, if there exists a spanning tree, there might be a non-planarity when representing the net by a star.

2.3 A Model for Components.

In order to clarify some of the points to be made, consider as an example a 16 pin dual-in-line package with 3 3-input gates as shown on Fig. 2.3.1.

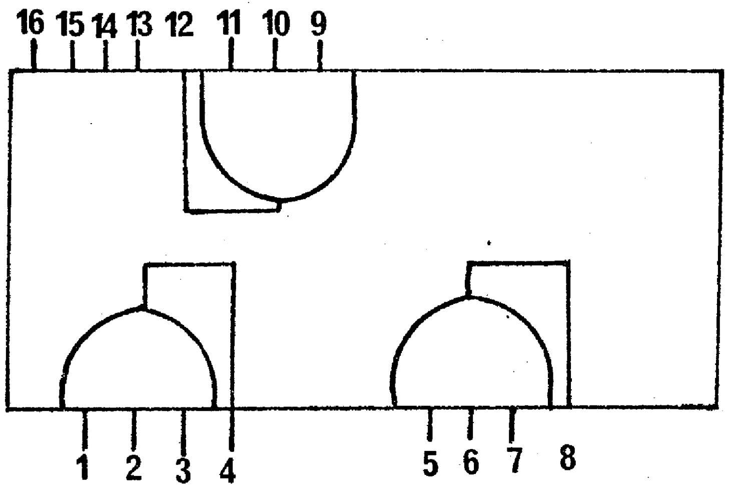


Fig. 2.3.1 Triple 3-input AND gate.

1) A module-to-cycle mapping is given in Fig. 2.3.2. This model allows wires to be located underneath the module only if they connect 2 or more pins of the same module; e.g. 4-11 . It is not possible to route connections between adjacent pins or longitudinally.

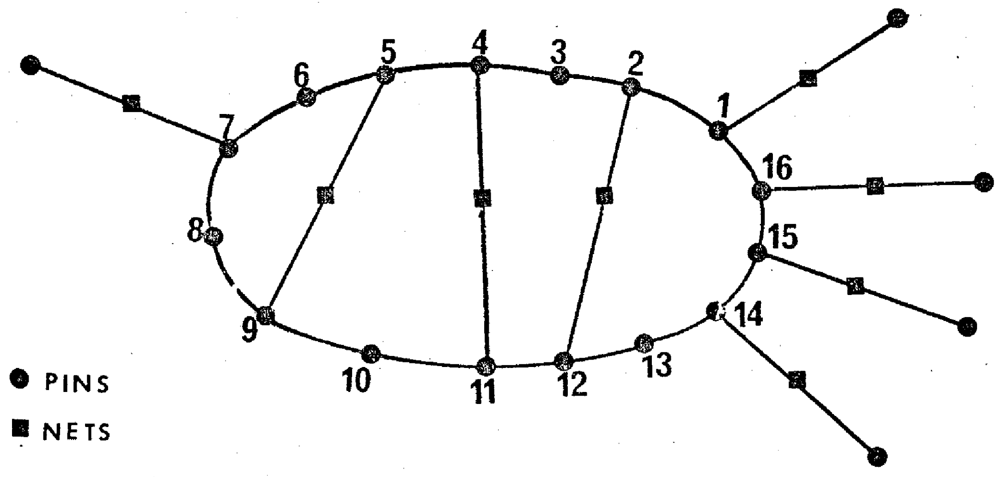


Fig. 2.3.2 Module to cycle mapping.

2) If no wires are to be allowed under the component, the region bounded by the cycle has to be divided such that no 2 pins belong to the same face, except when they are adjacent. An elegant way of doing this is to place a special vertex inside the region bounded by the cycle and to connect this vertex to all pin-vertices (this was proposed by VanLier and Otten (VO73)). This model is shown in Fig. 2.3.3.

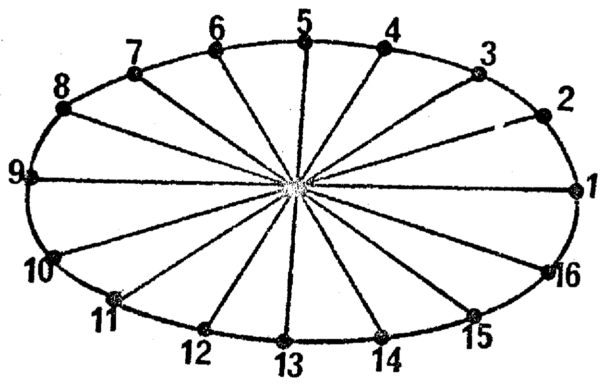


Fig. 2.3.3 Model to prevent wiring underneath a module.

3) In Fig. 2.3.1, pins 1,2,3 are logically equivalent and the same is true for 5,6,7 and 9,10,11. Instead of insisting on an ordering 1,2,3....,15,16 there is a partial ordering on the pins: (1,2,3),4,(5,6,7),8,(9,10,11),12,13,14,15,16.

Consider the model of Fig. 2.3.4: each group of n logically equivalent pins is represented by a star-like subgraph with the center vertex representing the group and the other vertices representing the pins. In the case n = 1 (no equivalent pins), the group is represented by a single vertex.

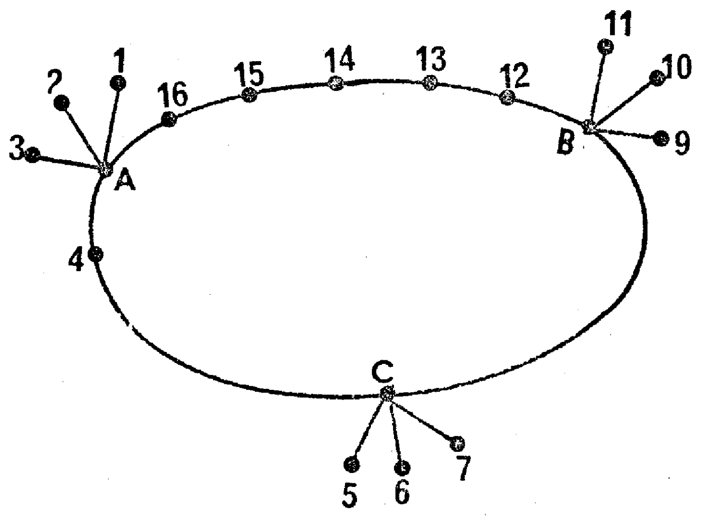


Fig. 2.3.4 Model that allows pins assignment.

The partial order relation is modelled by a cycle connecting the group in the order specified as is shown on Fig. 2.3.4. What are the advantages of this model? Suppose we use an algortihm to embed the graph in the plane; the result might be that the vertices appear in another order, e.g. 3,1,2 instead of 1,2,3. This means that the original (and arbitrary) choice of connecting net A to pin 1, net B to pin 2 and net C to pin 3 was not optimal and that we should have connected net A to pin 3, B to 1 and C to 2. In other words, this model allows pin assignment as a function of the embedding. It should be noted here that this model is valid only if logically equivalent pins are physically adjacent Fortunately, this is the case in many instances.

4) The 3 gates in the example are logically equivalent. Each of the sets of pins (1,2,3,4), (5,6,7,8), (9,10,11,12) is interchangable with another. This can again be formulated as as a partial ordering:
[(1,2,3,4),(5,6,7,8),(9,10,11,12)],13,14,15,16

If we take into account the equivalence of pins as in the previous model, this partial ordering becomes:
{[(1,2,3),4],[(5,6,7),8],[(9,10,11),12]},13,14,15,16

The model for this is shown in Fig. 2.3.5. The basic idea is that ordering is modeled by a cycle and equivalence by a star-like subgraph.
A theoretical justification for this model will be given in chapter 3. The partial ordering relation specifies that the set G, itself composed of the subsets G(1), G(2) and G(3), is to be followed by the

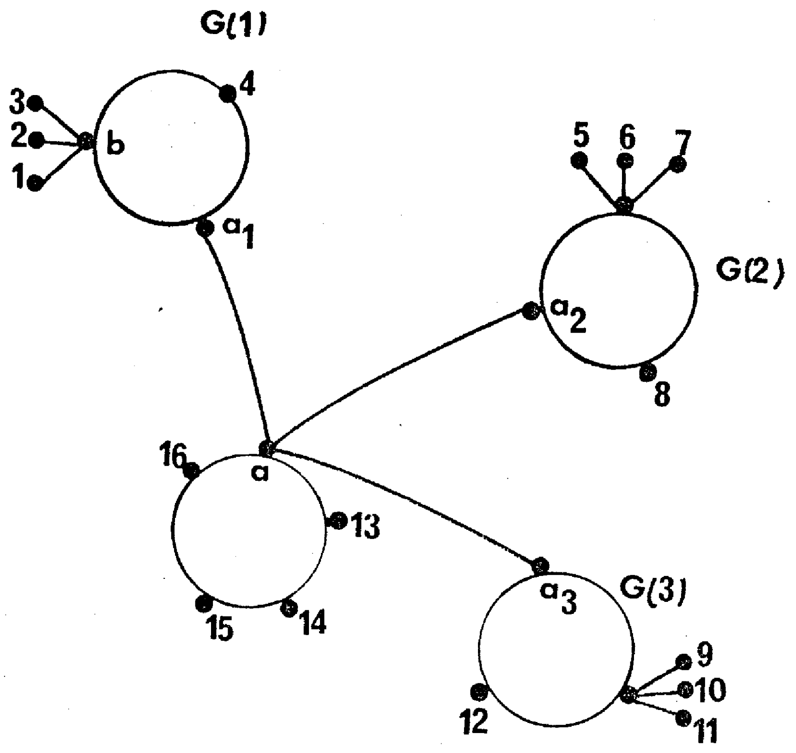


Fig. 2.3.5 Model that allows pin and gate assignment.

elements 13,14, 15 and 16 in that order. This is modeled by a cycle (a, 13, 14, 15, 16, a), where a is the center vertex of a star-like subgraph with the edges (a, a1), (a, a2) and (a, a3) connecting this center vertex to the subsets G(1), G(2) and G(3). Each of these subsets represents a single 3-input AND gate and is modeled as described before.

After using a graph embedding algorithm, the subgraphs G(1), G(2) and G(3) may appear in another order; e.g. G(3), G(1), G(2); this order gives us an optimal gate assignment as a function of the embedding.

This model is only valid when the pins of a subset (e.g. gate) are physically adjacent and when the equivalent subsets are physically adjacent. Physical adjacency means that there are no connected pins between 2 groups.

To illustrate this restriction, consider a triple 3-input AND gate with a pin configuration as shown in Fig. 2.3.6. Because of pin 5, none of the equivalent subsets are physically adjacent and therefore no gate assignment is possible.

In Fig. 2.3.7, an example is given of a configuration that allows partial gate assignment because the first two gates are physically adjacent.

An important remark to be made is that the cycles shown in these models are in reality directed cycles that may embedded with two different

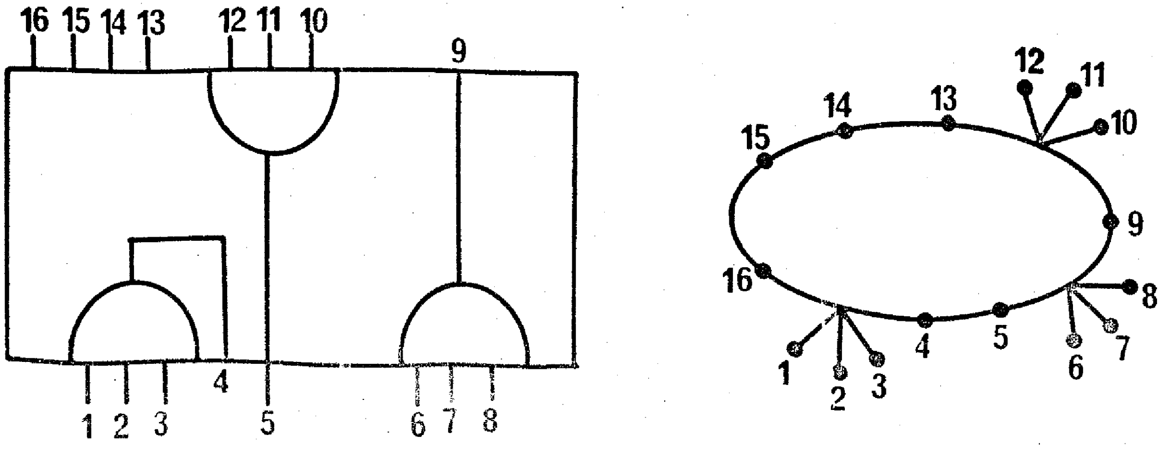
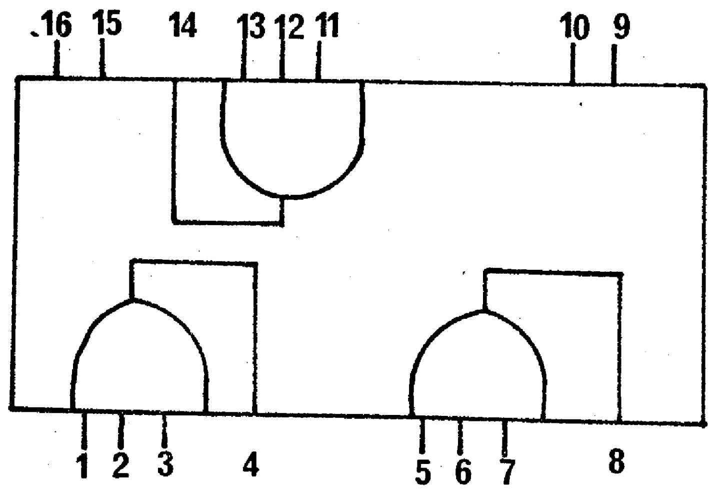


Fig. 2.3.6 Triple 3-input AND gate and its model with a pin configuration that does not allow gate assignment.

orientations. A cycle (1,2,3,4,1) could be embedded as (4,3,2,1,4) without this restriction; for a printed circuit board this would mean that the component would have to be placed on the other side of the board, which is not allowed in most cases. Therefore an embedding algorithm should take into account the prespecified orientation of the cycles.

5) For some components, the order in which the terminals appear is flexible and not defined a priori. An



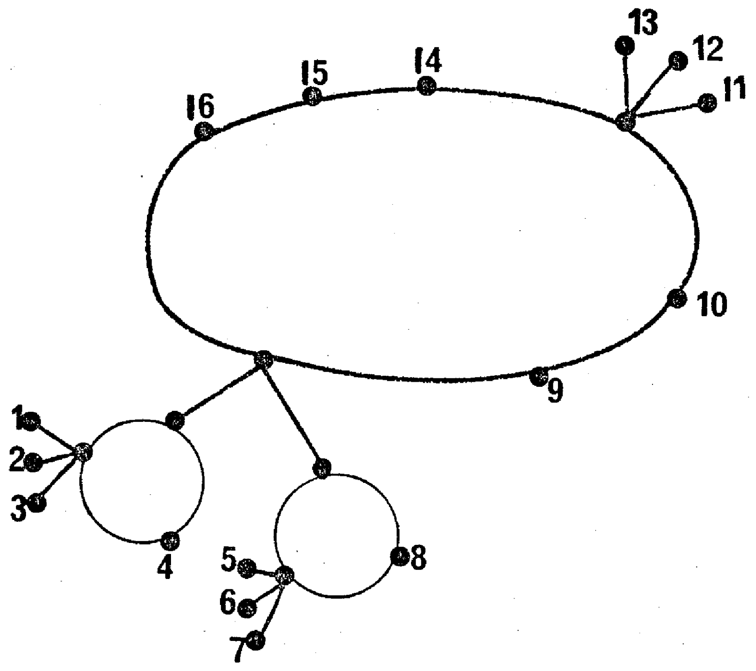


Fig. 2.3.7 Triple 3-input AND gate and its model with a pin configuration that allows partial gate assignment only.

appropriate model for such a device is a star-like subgraph.

6) Sometimes one or more terminals are physically equivalent. This means that a given net can be connected to either of a set of physically equivalent terminals.

As an example of physical equivalence of terminals, consider an IC transistor as in Fig. 1.3.12.a. Engl and Mlynski (EM73) model this by a star-like subgraph. It can be seen easily that this model is not capable of correctly representing the situation shown in Fig. 1.3.12.b and c, where three IC transistors are connected in parallel: the real circuit is planar while the model is non-planar.

A more correct model for an IC transistor can be derived in the following way: topologically, an IC transistor is a 6-terminal device (Fig. 2.3.8.a) with opposite terminals (1 and 6, 2 and 5, 3 and 4) being physically equivalent. The model is derived as follows:

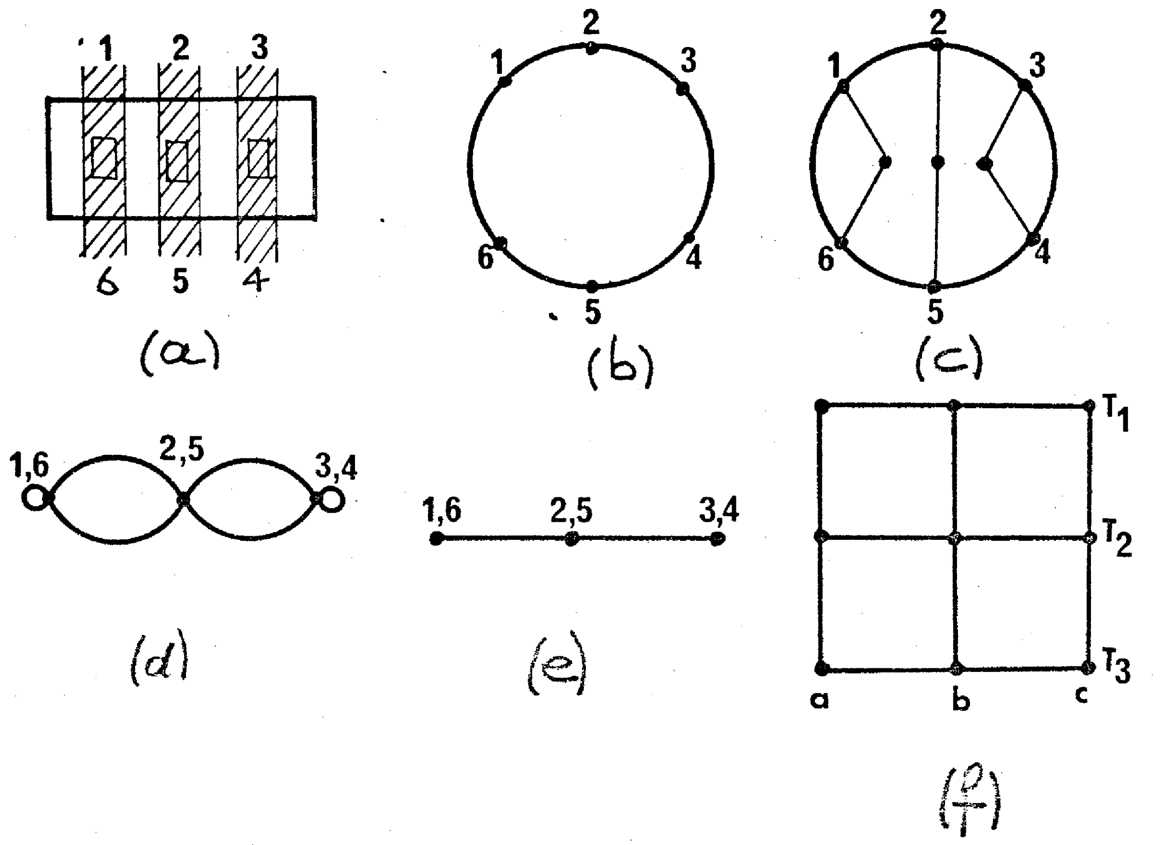


Fig. 2.3.8 (a) IC Transistor. (b) Model for the Order of the Terminals. (c-e) Derivation of the Model for Physical Equivalence. (f) Improved Model for the Circuit of Fig. 1.3.12.c.

1) draw a cycle, modelling the physical order of the 6 terminals (Fig. 2.3.8.b).

2) Connect the physically equivalent terminals by a star graph (Fig. 2.3.8.c). This is allowed only when none of these stars intersect.

3) Contract each star subgraph into a single vertex (Fig. 2.3.8.d).

4) Delete loop-edges and replace multiple edges by a single edge (Fig. 2.3.8.e).

This model yields a truly planar representation for the circuit of Fig. 1.3.12.c, as shown in Fig. 2.3.8.f.

This property can be used for performing pin assignment as a function of the layout.
Fig. 2.3.9.a shows, as an example, a 9-terminal component with terminals 3, 6 and 9 being physically equivalent. The model derived for this component is shown in Fig. 2.3.9.b. Assume that one of these physically equivalent terminals has to be connected to net A. The embedding algorithm can produce three different placements for net A. These are labeled A1, A2 and A3 on Fig. 2.3.9.b. The net would be

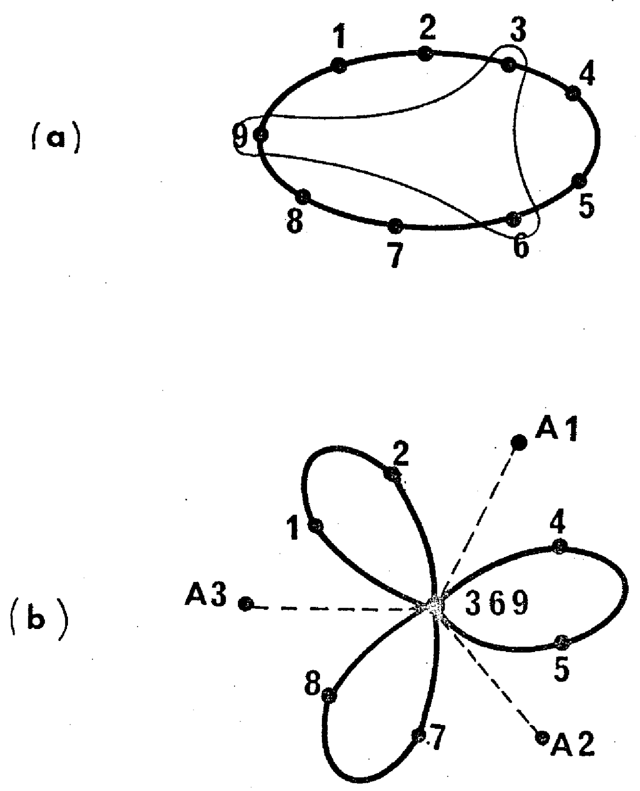


Fig. 2.3.9 (a) component with terminals 3, 6 and 9 physically equivalent. (b) model for the physical equivalence.

assigned to pin 3, 6 or 9 respectively, depending upon the result of the embedding algorithm.

2.4 A Model for Outside Connections.

If the order of the external connections is well-defined, then a proper model is a cycle, connecting all terminals in the order specified. It should be emphasized here that the graph embedding algorithm should position this cycle on the periphery.

Consider, as an example, a set of 20 external connections of which 1,2,3,4,9,10,18,19,20 are pre-assigned, while the other terminals can be assigned as a function of the embedding. For the 11 remaining signals, we have the following groups of terminals available: 5-8 and 11-17. The model proposed here requires us to partition the signals into 2 sets of 4 and 7 signals respectively. The model will allow pin assignment as a function of the embedding within each group. Once again we have a partial ordering:

{1,2,3,4,(5,6,7,8),9,10,(11,12,13,14,15,16,17),18,19,20}

As in the model for components, this ordering can be modeled by a cycle, where simple elements are represented by a vertex and groups by a star-like subgraph with the

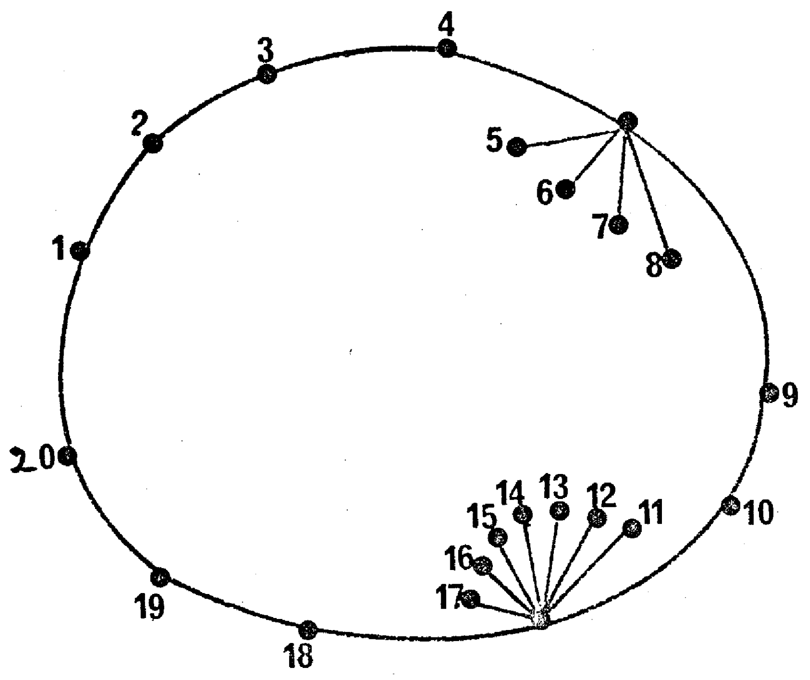


Fig. 2.4.1 Model for an outside connector.

center vertex on the cycle. The model for the example is shown in Fig. 2.4.1.

2.5 Summary of a Topological Layout Procedure.

It is appropriate here to summarize the main steps in a topological circuit layout procedure:

1- Construct a graph model from the circuit description

2- Use a constructive algorithm to embed the graph in the plane subject to the following constraints:

a) the cycle that models the outside connections should be on the periphery.

b) certain cycles that are representing component orientation should be embedded with the prespecified orientation.

3- If a one-layer layout is desired, a number of connections has to be removed in order to leave the graph planar. Note that only edges representing nets can be removed and not edges that model the components.

4- Making use of the technological properties such as wires between 2 adjacent pins or under the modules, try to route the remaining connections, taking into account finite wiring capacities.

5- Transform the topological layout into a physical layout by taking into account the real dimensions of the components.

3 Mathematical Models for the Circuit Layout Problem.

3.0 Introduction.

This chapter deals with the topological aspects of the circuit layout problem. Geometrical properties (e.g. the physical size and shape of the components) will be neglected.

A survey of previous work on mathematical models for the circuit layout problem is given in section 3.1. This survey complements the discussion of existing graph models, presented in chapter 1.

The mathematical structure of physical circuits is analyzed in section 3.2. The new concepts of physical equivalence of terminals and of logical equivalence of terminals and subcomponents are introduced.

Starting with the simple hypergraph model, used by Lawler (La73), these properties can be used to progressively refine this model.

A new and improved graph model for components is derived, in which the relationships that exist between the terminals of a component are used for a more optimal layout.

Some Graph Theoretical Definitions.

The definitions given here follow (BC71) and (Ha69), unless otherwise indicated.

A simple graph or briefly a graph is a system consisting of a finite non-empty set V (the vertices) and a family E of subsets of V of cardinality 2 (the edges). The vertex set of a graph G will be denoted by V(G) and the edge set by E(G).

Two vertices u and v are adjacent in a graph G if there exists an edge $\{u,v\}$ in E(G).

A graph H is a subgraph of a graph G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$.

A graph G with p vertices and q edges is embeddable on a surface S if it is possible to associate a collection of p distinct points on S (corresponding to the vertices of G) and a collection of q Jordan arcs (corresponding to the edges of G), such that if an arc "a" corresponds to an edge $e=\{u,v\}$, then only the endpoints of "a" correspond to vertices of G (i.e. u and v).

A graph G is planar if it can be embedded in the plane.

Two graphs are homeomorphic if both can be obtained from the same graph by successive subdivisions of edges.

The planarity of a graph can be characterized by some other property (Ha69) such as:

(Kuratowski) A graph is planar if and only if it does not contain a subgraph homeomorphic to either K(5) or K(3,3).

(Harary and Tutte) A graph is planar if and only if it does not have a subgraph contractible to K(5) or K(3,3).

(Maclane) A graph G is planar if and only if for every block of G with at least 3 vertices, there exists a cycle basis Z(1), Z(2), ..., Z(m) and one additional cycle Z(0), such that every edge belongs to exactly two of these cycles.

(Whitney) A graph is planar if and only if has a combinatorial dual.

A _plane graph_ is a graph that is embedded in the plane.

A _region_ or a _face_ of a plane graph G is a maximal portion of the plane for which any two points may be joined by a Jordan arc "b", such that any point of "b" neither corresponds to a vertex of G nor lies on a Jordan arc corresponding to an edge of G.

The _boundary_ of a region R of a plane graph consists of all the points x corresponding to vertices of G or lying on a Jordan arc corresponding to an edge of G such that x can be joined to a point of R by a Jordan arc all of whose points (except for x) belong to R.

A graph G is _outerplanar_ if it can be embedded in the plane such that every vertex of G lies on the boundary of some region (usually the exterior).

The following definitions concerning oriented graphs are adapted from (Ul70):

Let G(V,E) be a graph with a vertex set V and an edge set E. The _neighbberhood_ N(v) of a vertex v of G is the set of all vertices of G that are adjacent to v.

An _orientation_ o(v) of a vertex v of G is a cyclic permutation of the elements of the neighborhood of v.

An _orientation O(G)_ of a graph G is a mapping of the vertex set V into the set of orientations of all vertices of G.

The triple (V,E,O) is an _oriented graph_.

A graph G(V,E,O) is _partially oriented_ if the mapping of V into the set of orientations is partially defined (i.e. O(G) is defined for a proper subset of V only).

An _oriented vertex_ is a vertex for which an orientation is defined.

An _oriented graph G is planar_ if it can be embedded in the plane such that for the arcs a(i) with a common endpoint P, that correspond to the edges incident to a given vertex v, a clockwise sweep around P encounters these arcs a(i) in the order prescribed by the orientation. It should be noted that with every plane graph, one can associate an oriented graph. However, not every oriented graph has a corresponding plane graph (i.e. is planar).

Let a,b and c be three adjacent edges incident to a vertex v; by b >< [a,c], we will indicate that a clockwise sweep around v encounters these three edges in the order a, b, c (i.e. b lies between a and c).

A planar oriented graph is <u>outerplanar</u> if it can be embedded such that every vertex of G lies on the boundary of some region (usually the exterior region).

<u>G is an oriented graph of type 1</u> if there exist in G two distinct vertices m and n and three paths p_1, p_2 and p_3, from n to m, such that each edge of G belongs to exactly one of these paths and if N_1, N_2 and N_3 are edges incident to n and belonging to p_1, p_2 and p_3 respectively and if M_1, M_2 and M_3 are edges incident to m and belonging to p_1, p_2 and p_3 respectively then $N_1 >< [N_2, N_3]$ if and only if $M_1 >< [M_2, M_3]$. An example of such a graph is given in Fig. 3.0.1a.

<u>G is an oriented graph of type 2</u> if there exists a vertex n and two cycles C_1 and C_2 such that each edge of G belongs to exactly one of the cycles and if vertex n is incident to edges M_1 and N_1 of cycle C_1 and to edges M_2 and

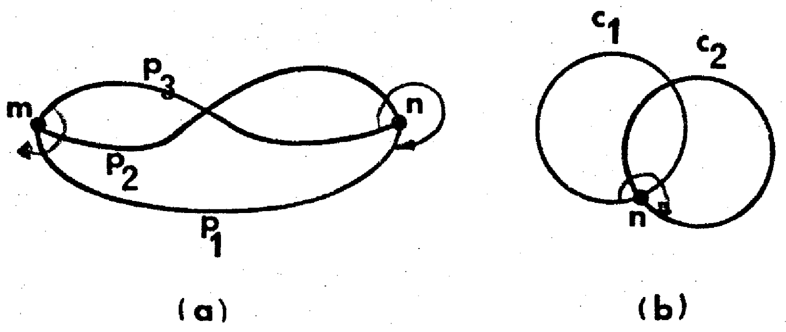


Fig. 3.0.1 (a) Oriented graph of type 1. (b) Oriented graph of type 2.

N_2 of cycle C_2 then M_2 >< $[M_1,N_1]$ if and only if M_1 >< $[M_2,N_2]$. Fig. 3.0.1b shows an example.

<u>Theorem</u>: if G is a minimal non-planar subgraph of an oriented graph , then G is an oriented graph of type 1 or G is an oriented graph of type 2 (U170).

For two oriented graphs G and G', a mapping α from V(G) onto V(G') is called an <u>elementary contraction</u> if there are two adjacent vertices v_1 and v_2 in G such that

(1) $\alpha v_1 = \alpha v_2$.

(2) $(w_1,w_2) \in E(G)$ iff $(\alpha w_1,\alpha w_2) \in E(G')$ provided that w_1 and w_2 are distinct from v_1 and v_2.

(3) for $w \in V(G)$ and distinct from v_1 and v_2, (v_1,w) or $(v_2,w) \in E(G)$ iff $(\alpha v_1,\alpha w) \in E(G')$.

(4) let x_1,x_2,y_1,y_2 denote (possibly empty) sequences of vertices and let (x_1,v_2,y_1) be the sequence of adjacent vertices of v_1 encountered in a clockwise sweep and let (x_2,v_1,y_2) be the same sequence for v_2. Then the sequence for the new vertex $\alpha v_1 = \alpha v_2$ is $(\alpha x_1,\alpha y_2,\alpha x_2,\alpha y_1)$. An example is given in Fig. 3.0.2.

In Fig. 3.0.2 (a) and (c), two oriented graphs are shown, while their contractions are depicted in (b) and (d) respectively. For the first graph (Fig. 3.0.2.a), the sequences are: $x_1=c_1$; $y_1=b_1$; $x_2=c_2$ and $y_2=b_2$. The contraction of edge (a_1,a_2) results in the sequence (c_1,b_2,c_2,b_1) of the adjacent vertices around vertex a. For the second graph (Fig. 3.0.2.b), the sequences are: $x_1=b_1$; $y_1=c_1$; $x_2=b_2$ and $y_2=c_2$. After contracting (a_1,a_2) the sequence around vertex a is (b_1,c_2,b_2,c_1).

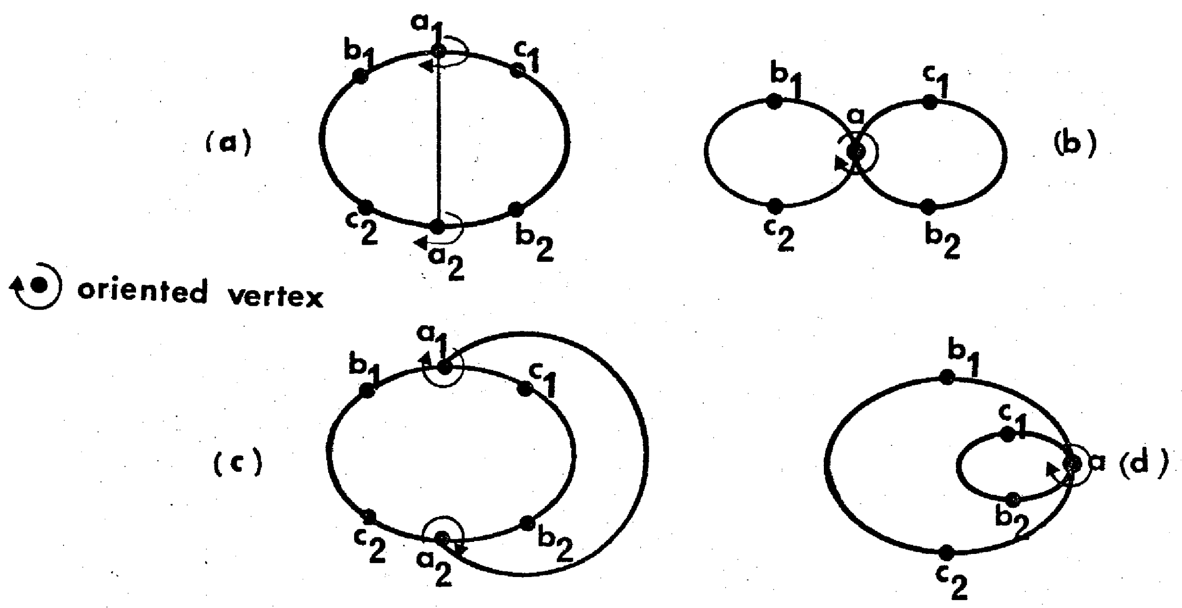


Fig. 3.0.2 Planar oriented graphs (a,c) and their contractions (b,d).

Let G be an oriented graph, outerplanar with respect to a vertex set U. Then an elementary contraction $G' = \alpha(G)$ is outerplanar with respect to $U' = \alpha(U)$.

3.1 A Survey of previous Mathematical Formulations.

In chapter 1, graph-theoretical models for the circuit layout problem were critically analyzed. In this section, other existing abstract models for the problem are discussed.

3.1.1 Multiplace Graphs.

The mathematical model for the circuit layout problem, introduced by Engl and Mlynski (EM75), is based on the concept of <u>multiplace graphs</u>.

Let A be a set of cardinality m; then a <u>multiplace relation</u> † R on the set A is defined as:

$$R \subseteq \bigcup_{i=1,m} Ai$$

where Ai is the set of all multisets ‡ of cardinality i, defined on the set A. Then R is a family of multisets of cardinality less than or equal to m, defined on A.

† The use of the term 'relation' in the definition of multiplace relation and n-place relation (EM75, Def.1) is rather confusing, since the elements of a relation are commonly defined as ordered pairs (or n-tuples) in the mathematical literature (e.g. Ha60), while in (EM75), the elements of a multiplace relation are unordered n-tuples (in particular multisets).

‡ A multiset is also known as a weighted set.

A <u>multiplace graph</u> is a pair G(A,R), where A is a finite set of vertices and R is a multiplace relation on A. Each element of R is an edge of the multiplace graph.

Engl and Mlynski proposed the following model for the circuit layout problem:

- Nets are represented by vertices of the multiplace graph.

- A n-terminal component is modelled by a n-ary relation, called <u>spider with n legs</u> †

This multiplace graph is called the <u>potential graph</u> of the circuit. One can define an incidence matrix M for a multiplace graph with elements m(i,j),

where $m(i,j) = 0$ if $a(i) \notin r(j)$

$= n$ if $a(i) \in r(j)$ and occurs n times.

The transpose of the multiplace graph G(A,R) is another multiplace graph G'(A',R') which has as its incidence matrix M' the transpose of the original incidence matrix M. The transpose of the previously mentioned potential graph is called the <u>component graph</u>. Here the

† An n-legged spider is a multiset of cardinality n.

components are represented by vertices and the nets by 'spiders'.

In the potential graph, a component with n pins is modelled by a multiset of cardinality n ("a spider with n legs"). The reason why multisets are needed to model a component is that a component can be connected more than once to the same net. For the same reason multisets are needed to model nets in the component graph. Since the only elements modelled by Engl and Mlynski are components and nets, it is not possible to represent the relationships that exist between the pins of a component.

Engl and Mlynski introduce the concept of planarity of a multiplace graph in (EM75). They first define a mapping of a multiplace graph M(A,R) into a simple graph G(V,E): every element of the set A is mapped into a distinct vertex v of the graph G. Every n-place relation of R is mapped into a K(1,n) (i.e. star-) subgraph, with the center vertex representing the n-place relation and the edges representing the fact that the vertices of A belong to the n-place relation.

This mapping can result in a graph with multiple edges. A multiplace graph is planar if the corresponding simple graph is planar. Since multiple edges do not influence planarity, they can be replaced by a single edge. This results in the model described in section 1.3.4.

Multiplace graphs do not take into account many of the properties of circuits, that are discussed in section 3.2. Furthermore, it is shown in section 3.3 that the existing concept of hypergraph is appropriate for modelling those properties of physical circuits, that can be modelled by multiplace graphs.

3.1.2 Hypergraphs

Consider the following definition from (Be72):

A hypergraph H(V,E) is an algebraic structure where V is a set of elements, called vertices and E is a family of subsets E(i) of V. These sets E(i) are called hyperedges. Each set E(i) is a non-empty subset of V. Furthermore

$$\bigcup_{i=1,|E|} E(i) = V$$

In (La73), it is mentioned that for solving the partitioning problem, a circuit can be represented by a hypergraph, with the vertices representing the components and the hyperedges representing the nets. The problem can then be formulated as the partitioning of a hypergraph into subgraphs with vertex sets of cardinality not greater than a given constant k, such that the number of hyperedges connecting different subgraphs is minimized. Although this model is very similar to the component graph, mentioned by Engl and Mlynski, it was never used in conjunction with the circuit layout problem.

3.2 Mathematical Properties of Physical Circuits.

In this section, the mathematical structure of physical circuits is studied. New concepts of physical and logical equivalence of terminals and components are introduced.

3.2.1 A Simple Model for Components and Nets.

Let $\tilde{C} = \{ C(i) \}$ be the set of components.

Every component has a number of terminals † to which interconnections are made. A net is a collection of terminals (of the same or of different components), that are equipotential at all times. The terminals of a net are made equipotential by connecting them by a continuous strip of conducting material.

When one does not want to distinguish between the different terminals of a component, the nets can be defined as a family $\tilde{N}$ of subsets $N(i)$ of $\tilde{C}$.

† The terms 'terminal' and 'pin' will both be used for the physical locations of a component to which interconnections are made.

A physical circuit can then be modelled by a system $(\tilde{C},\tilde{N})$, where $\tilde{C}$ is the set of components and $\tilde{N}$ is a family of subsets of $\tilde{C}$, i.e. the nets.

For solving the circuit layout problem it is often desirable to distinguish between the different terminals of a component since this can lead to a better layout.

A component can be considered as a set C(i) of terminals p(i,j). Let P be the set of all terminals. Note that $C(i) \bigcap C(j) = \emptyset$ for $i \neq j$ and

$$\bigcup_{i=1,|\tilde{C}|} C(i) = P$$

The set of components is now a family $\tilde{C}$ of subsets of P. We define a net as a set of terminals that are connected to each other. Logically, there is no sequence in which the terminals of a net should be connected. Therefore a set is an appropriate model for this relationship.

Consider the set P of all terminals: between two terminals that that belong to the same net, there exists a relation "are interconnected". This relation is reflexive, symmetric and transitive and therefore is an equivalence relation. This equivalence relation partitions the set P into a number of disjoint subsets (equivalence classes), called nets. A net N(k) is a subset of P. Again $N(i) \bigcap N(j) = \emptyset$ for $i \neq j$.

When a terminal p is not connected to any other terminal, then it belongs to a net N of cardinality 1. Such nets are degenerate. Proper nets are of cardinality 2 or greater. The total wiring $\vec{N}$ (set of all nets) is family of subsets of P. A physical circuit can then be modelled by a triple $(P,\vec{C},\vec{N})$.

3.2.2 Relations between the Terminals of a Component: Cyclical Ordering.

In many cases, the order in which the terminals of a component appear on the periphery of this component is predefined. This can be represented by defining a function S(i), called the <u>successor function</u>, for each element of the set C(i); this function S(i) maps an element of C(i) into another element of C(i). This mapping is one-to-one. The inverse function R(i), called the <u>predecessor function</u>, also maps every element of C(i) into another element of C(i).

3.2.3 Relations between the Terminals of a Component: Logical Equivalence of Terminals.

It often occurs that a number of terminals of a component have identical logical functions; this allows terminals to be interchanged in order to obtain a better layout. An example of such a situation is an AND gate with 3 inputs: suppose that each of the terminals has to be connected to a net, one should not a priori assign a net to a physical terminal but rather do this in function of an optimal layout. This problem is often referred to as the pin assignment problem.

Let L = { p(j) } be a subset of C(i), consisting of terminals having identical logical functions. Each of the terminals p(j) is incident with a net N(i). Since the terminals in the set L all perform identical logical functions, it is permissible to assign the nets N(i) to any permutation of the terminals p(j). The terminals in the set L are logically equivalent.

This situation can be modelled as follows: let C(i) = { p(i,j) } be the set of terminals representing component i, as defined in section 3.2.1. Each group of logically equivalent terminals can be represented by a set L(i,j) of terminals. The cardinality of each such set is at least 1. We can represent component i by a set C(i) = { L(i,j) }.

3.2.4 Relations between the Terminals of a Component: Physical Equivalence of Terminals.

By <u>physical</u> <u>equivalence</u> of a set of terminals of a component is meant that all terminals in this set are equipotential. This means that a net can be connected to any one of these terminals. +

As an example of this, consider an IC transistor: topologically, one can consider this as a 6-terminal component, with opposite terminals being physically equivalent, as shown in Fig. 3.2.1.

+ It is also possible to split the net into several subnets, each of them connected to one of the physically equivalent terminals. Being equipotential assumes that there is an interconnection between these terminals within the component. By using the model proposed here, the net will be connected to only one of the physically equivalent terminals in function of a planar layout. If the net is not completely embedded, then one can attempt to split the net into subnets as indicated in section 5.4.

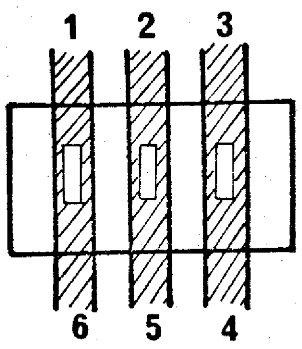


Fig. 3.2.1 Physical equivalence of the terminals of an IC transistor.

Let $F = \{ p(j) \}$ be a subset of $C(i)$, consisting of terminals $p(j)$, at least one of which has to be connected to a net $N(k)$. The terminals in the set F will be called <u>physically equivalent</u>.

This property can be modelled as follows: Each group of physically equivalent terminals of component i can be represented by a set $F(i,j)$, with $|F(i,j)|$ at least equal to 1. Component i can then represented by the set $C(i) = \{ F(i,j) \}$.

It is possible that within a component both physical and logical equivalence of terminals exists. In that case, the following property holds: if $p \in F(i,j)$ and $p \in L(i,j)$, then for all elements $q \in F(i,j)$, $q \in L(i,j)$.

This property allows us to represent the component in the following way: C(i) = { L(i,j) }, where L(i,j) = {F(i,j,k) }.

3.2.5 Relations between the Terminals of a Component: Subcomponent Equivalence.

In some cases, a component can be made up of a number of identical subcomponents, that are logically interchangeable. An example of this is a quadruple 2-input NAND gate, which is a common IC module available commercially. Another example is a 3-3-3 AND-OR-INVERT gate (Fig. 3.2.2).

In such cases, one should not randomly assign a physical subcomponent to a particular group of nets to be connected to one such subcomponent, but rather do this in function of an optimal layout.

This can be modelled by representing component i by a set of equivalent subcomponents E(i,j): C(i) = { E(i,j) }, where E(i,j) = { L(i,j,k) }, i.e. each subcomponent consists of a set of logically equivalent terminal sets L(i,j), then L(i,j,k) = { F(i,j,k,l) } i.e. each set of

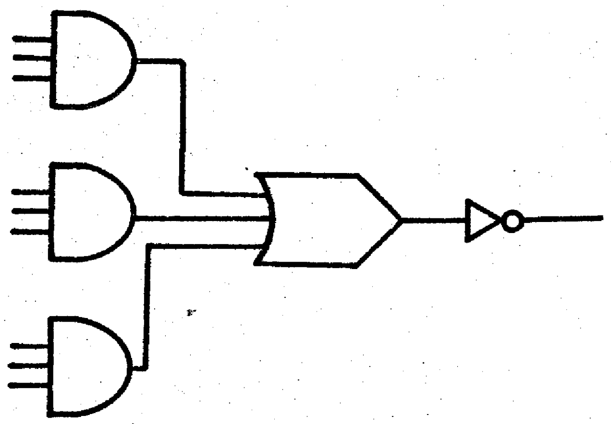


Fig. 3.2.2 A 3-3-3 AND-OR-INVERT gate as an example of subcomponent equivalence.

logically equivalent terminals consists of a set of physically equivalent terminal sets, and finally, $F(i,j,k,l) = \{ p(i,j,k,l,m) \}$ i.e each set of physically equivalent terminals consists of a number of simple terminals.

The relations between the terminals of a component, as described in sections 3.2.3 to 5, form a hierarchy, as shown in Fig. 3.2.3.

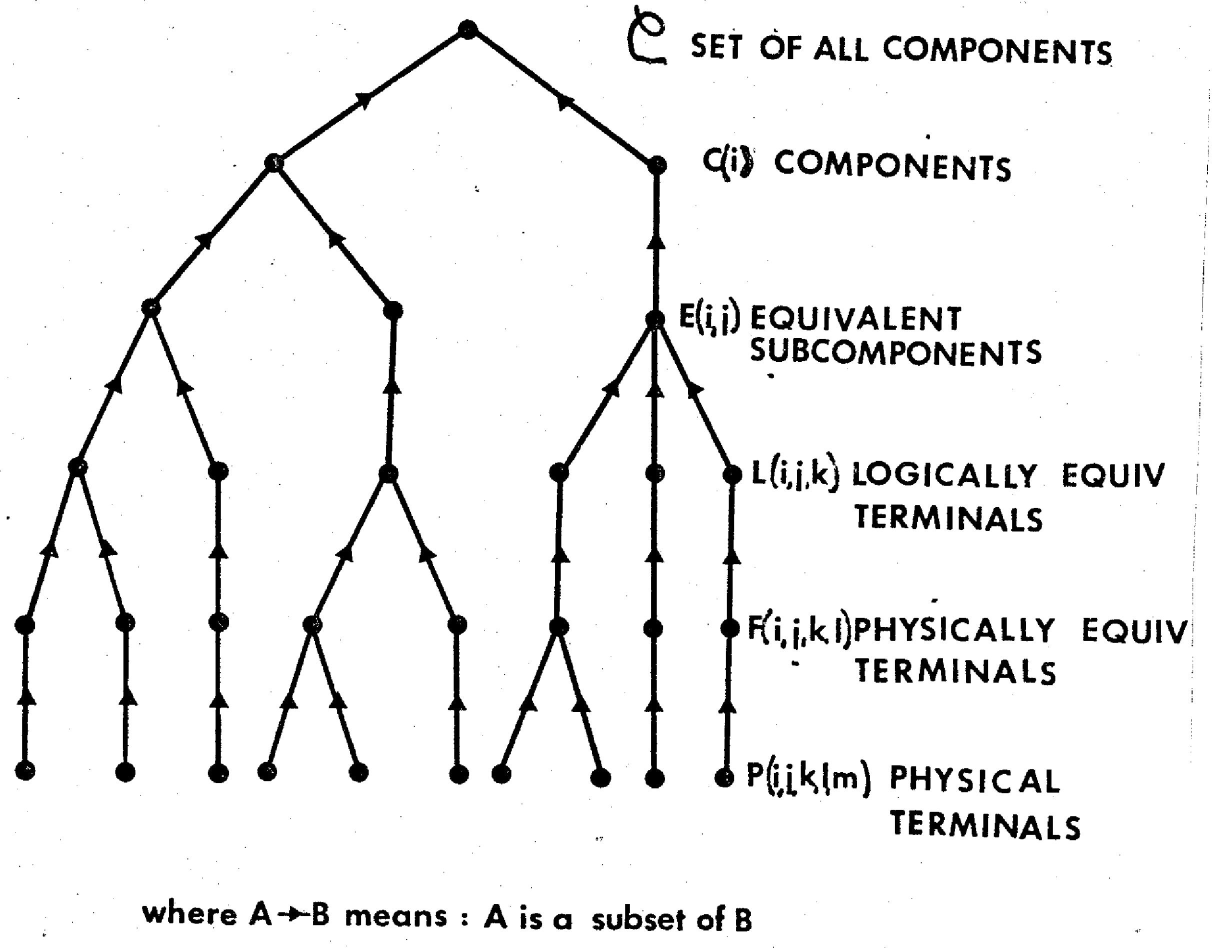

$\mathcal{C}$ SET OF ALL COMPONENTS
C(i) COMPONENTS
E(i,j) EQUIVALENT SUBCOMPONENTS
L(i,j,k) LOGICALLY EQUIV TERMINALS
F(i,j,k,l) PHYSICALLY EQUIV TERMINALS
P(i,j,k,l,m) PHYSICAL TERMINALS
where A→B means : A is a subset of B

Fig. 3.2.3 Hierarchy of relations between the terminals of a component.

3.3 Hypergraphs.

3.3.0 Introduction.

In section 3.3 we will study the use of hypergraphs as a mathematical model for the circuit layout problem. First a simple model will be discussed that is sufficient for partitioning a physical circuit. This simple model will then be expanded progressively, thereby taking into account more of the properties of physical circuits in each refinement. Finally, some consideration will be given to defining the planarity of hypergraphs, as related to the circuit layout problem.

3.3.1 A Simple Hypergraph Model.

In section 3.2.1, the circuit layout problem was modelled by a system $(\tilde{C},\tilde{N})$, where $\tilde{C}$ is the set of components and $\tilde{N}$ is a family of subsets of $\tilde{C}$, representing the nets.

From the definition of a hypergraph, it is obvious that a simple model for the circuit layout problem is a hypergraph, where the set of components maps into the set of vertices and the sets N(k), i.e. the nets, form the hyperedges. In section 3.3.4, mappings of hypergraphs into simple graphs for the purpose of defining planarity will be discussed.

As already mentioned in section 3.1.2, this simple hypergraph model can be used for the circuit partitioning problem.

3.3.2 A more Detailed Hypergraph Model.

Consider a hypergraph, where the set P of terminals is represented by the set of vertices and the sets C(i) and N(k) are the hyperedges. Using Berge's method of representing hypergraphs, Fig. 3.3.1 shows a circuit and its hypergraph model.

Since it is important to distinguish between the different terminals of a component, this hypergraph model is more appropriate for the circuit layout problem than multiplace graphs. Since a terminal belongs to exactly one component and to at most one net, there is no longer a need for multisets.

The use of sets to model components is not sufficient because there exist a number of interesting relationships between the terminals of a component that can be used in solving the layout problem.

It should be noted that the sets $\tilde{C}$ and $\tilde{N}$ both are partitions of the vertex set of the hypergraph.

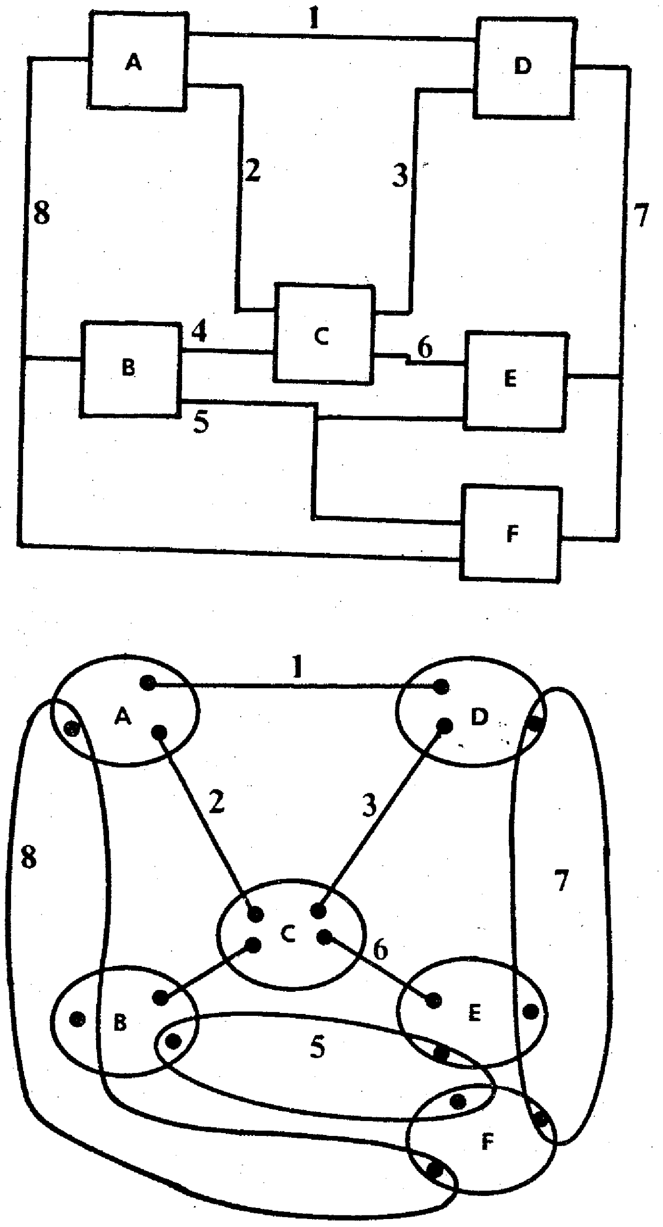


Fig. 3.3.1 A Circuit and its Hypergraph Model.

3.3.3 Oriented Hypergraphs.

It was mentioned in section 3.2.2 that there exists a cyclical relationship between the terminals p(i,j) of a component C(i). This can be written as a relation R:

$$p(i,1)\ R\ p(i,2)\ R\ p(i,3)\ \ldots\ldots\ldots\ R\ p(i,n)\ R\ p(i,1)$$

The introduction of this relation between the terminals of a component makes the previous model no longer a proper hypergraph. It is possible however to redefine the model here, such that the graph representing the circuit remains a hypergraph. Let the set of terminals P be the set of vertices of the hypergraph. Each net N(k) is represented by a hyperedge. Instead of representing the sets C(i) by hyperedges, we represent it by a set K(i) of simple directed edges, representing the cyclical nature of the relationship. The set of terminals corresponds to the set of vertices of the hypergraph.

The set K(i) then consists of the directed edges [p(i,1),p(i,2)] , [p(i,2),p(i,3)] , ... [p(i,n),p(i,1)] The algebraic structure $(V,\tilde{N},K)$, where V is a set of vertices representing the component terminals, $\tilde{N}$ is a set of hyperedges, representing the nets and K is a set of simple directed edges, representing the component boundaries, will be called an <u>oriented hypergraph</u>. Fig. 3.3.2 shows the oriented hypergraph model for the circuit of Fig. 3.3.1.

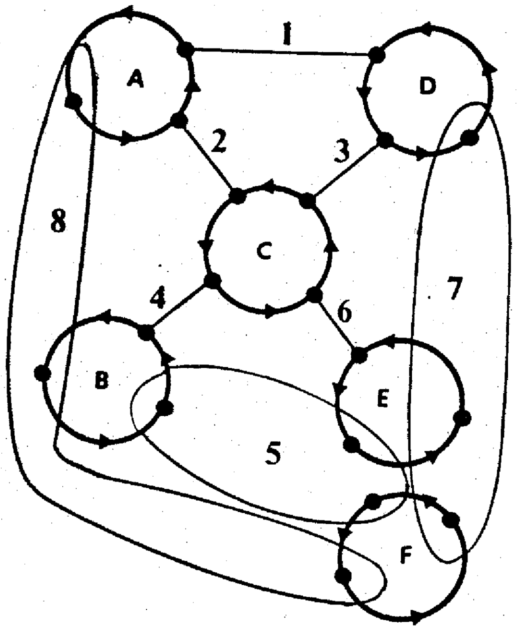


Fig. 3.3.2 Oriented Hypergraph Model representing the Cyclical Ordering of Terminals.

3.3.4 Other Relations between Component Terminals.

It is possible to continue expanding this hypergraph model of the previous section, thereby taking care of the properties mentioned in sections 3.2.3 to 3.2.5.

In section 3.4, simple † graph models are derived for components. Nets, however, remain true sets and cannot always be represented correctly by simple graphs.

3.3.5 Planarity of Hypergraphs.

In (EM75), it is stated that the representative graph of a hypergraph, as defined by Berge (Be72), is not an adequate model for planar circuit layout because it contains a K_5 subgraph for every node of the hypergraph with a degree ≥ 5. This does not imply that a hypergraph is not an appropriate model for the circuit layout problem since Berge's representative graph is only one of many simple graphs into which a hypergraph can be mapped.

In particular, one could define the following mapping of a hypergraph $H(A,X)$ into a simple graph $G_1(V_1,E_1)$:

† A simple graph is defined in section 3.0.

a) every vertex a of H is mapped into a vertex v' of G_1 [$v'=f(a)$]; let V_1' be the set of all such vertices of G_1.

b) for every hyperedge x of H, there is a vertex v'' in G_1 [$v''=g(x)$]; let V_1'' be the set of all such vertices of G_1.

c) for every vertex a in a hyperedge x, there is an edge (v_1,v_2) in G_1, such that $f(a)=v_1$ and $g(x)=v_2$. Furthermore, $V_1' \cap V_1'' = \emptyset$ and $V_1' \cup V_1'' = V_1$.

This mapping is similar to the one for multiplace graphs, given in (EM75, def. 9).

We can then define a <u>hypergraph</u> to be <u>planar</u> (<u>first definition</u>) if and only if its associated simple graph G_1 is planar.

This definition of hypergraph planarity can lead to difficulties with respect to the circuit layout problem. Assume that components are modelled by simple subgraphs and nets by proper hyperedges (of which the cardinality can be greater than 2). Then the mapping defined here is not adequate since a net can be realized as any spanning tree on its vertex set.

We can define a second mapping of a hypergraph $H(A,X)$ into a simple graph $G_2(V_2,E_2)$ as follows:

a) Let $V(G_2) = A(H)$

b) Every hyperedge x of H is mapped into a spanning tree on the set of vertices $\{v(i) \mid v(i) = a(i) \text{ and } a(i) \in x\}$.

We now define a <u>hypergraph</u> to be <u>planar</u> (<u>second definition</u>) if and only if there exists an associated simple graph G_2 (as defined above), that is planar.

<u>Theorem</u> If a hypergraph is planar, according to the first definition, then it is planar according to the second definition.

<u>Proof:</u> Let H be a hypergraph, planar according to the first definiton of planarity and let G_1 be its associated simple graph. Then G_1 is planar.

A hyperedge x is mapped into a $K(1,n)$ subgraph of G_1. Let v be the center vertex and $v(i)$, $i = 1,n$

where $v(i) = f(a(i))$, the end vertices of this star-subgraph. Apply the following transformation to G_1: select an edge $(v,v(i))$ of the star-subgraph and contract it. The resulting graph is still planar. † We can repeat this procedure for all hyperedges of H.

The resulting simple graph G_2 satisfies the mapping used to define the planarity according to definition 2. Since G_2 is planar, H is planar according to definition 2.

The converse of this theorem is not true.

For solving the circuit layout problem optimally, it is necessary to use the second definition of hypergraph planarity. This would require enumerating all spanning trees over all of the hyperedges. Since this is impractical, the less desirable definition of hypergraph planarity will be used in conjunction with the circuit layout problem.

----

† A graph G is planar iff it does not have a subgraph contractible to K(5) or K(3,3). (Ha69).

3.4 Graph Models for Components.

3.4.1 Desirable Properties of an Adequate Component Model.

1) In order to avoid introducing non-planarities the component model should be outerplanar † with respect to the vertices, that represent the component's terminals.

2) Every possible embedding of the component together with the nets connected to it should be compatible with the cyclical ordering of the terminals on the component's boundary, if such ordering is prescribed. In other words, every possible embedding must have a physical meaning.

It will be assumed here that an n-terminal net is modelled by a K(1,n) subgraph. Furthermore, it will be required that nets be embedded in the region exterior to a component's boundary.

† A graph is outerplanar if it is planar and if it can be embedded in the plane, such that all its vertices lie on the same face (usually the exterior face) (Ha69). Let G(V,E) be graph and let V_1 be a subset of V, then G is outerplanar with respect to V_1 if it can be embedded in the plane such that all vertices of V_1 lie on the same face.

3.4.2 Cyclic Ordering of the Terminals.

In section 3.2.2, it was mentioned that the ordering of terminals along a component's boundary can be modelled by a successor function. In section 3.3.3, this cyclic relationship was represented by a directed cycle, connecting the terminals.

When we assume nets to be embedded in the region exterior to the component's boundary and if the nets are represented by K(1,n) subgraphs, then we can model a component and its incident net-edges by a partially oriented graph G, as shown in Fig. 3.4.1(a). The vertex set of G consists of oriented vertices, representing terminals and of non-oriented vertices, representing the nets. The edge set of G consists of a cycle, modelling the component's boundary and of edges, connecting the terminals to the net-vertices.

In certain cases, a mirror-image component is available. Since the orientation of the component's boundary is not predefined in this case, two different embeddings are possible and then oriented graphs cannot be used.

In this case, an undirected cycle is an appropriate model. In order to avoid the embedding of connections inside the cycle, a star-subgraph should be placed inside the cycle.

3.4.3 Physical Equivalence of Terminals.

Let $C = \{F(i)\}$, where $F(i)$ is a set of physically equivalent terminals. $|F(i)| \geq 1$ and $F(i) \cap F(j) = \emptyset$ if $i \neq j$

Let $C' = \{p(j)\}$ be the set of terminals. Then the family $\tilde{F}$ of sets $F(i)$ forms a partition of C'.

It will be assumed here that no two elements of $F(j)$ are physically adjacent. If two or more terminals are physically adjacent, there is no problem of pin assignment between them and they can be represented by a single terminal.

Assume that the cyclic ordering of the terminals is given by a successor function. This can be written as a relation R: $p(1)\ R\ p(2)$, $p(2)\ R\ p(3)$,....... $p(n)\ R\ p(1)$. This successor function induces the following relation R' on the elements of the family $\tilde{F}$: for all $p(i)$, $p(j)$ such that $p(i)$ R $p(j)$ with $p(i) \in F(k)$ and $p(j) \in F(l)$ (where $F(k) \neq F(l)$): $F(k)$ R' $F(l)$

First assume that no mirror-image components are available. Consider the partially oriented graph G', obtained from the original graph G by contracting the sets of physically equivalent terminals. The vertex set of G' consists of vertices, representing the sets of physically equivalent terminals and of vertices, representing the nets. The edge set of G' consists of the edges, connecting the terminals to the corresponding net-vertices and of edges $\{a,b\}$, for which the relation a R' b holds. An example of this is given in Fig. 3.4.1(b), where the sets X and Y are contracted.
Note that the original cycle may be replaced by a number of cycles and that the resulting graph may be 1-connected.

In order for the resulting model to be suitable, it has to be outerplanar for every possible embedding that has a physical meaning. Furthermore, a vertex corresponding to a set $F(j)$, of cardinality greater than 1, must be non-oriented, in order to allow the incident net-edge to be embedded properly. The vertices, corresponding to sets $F(j)$ of cardinality 1, remain oriented.

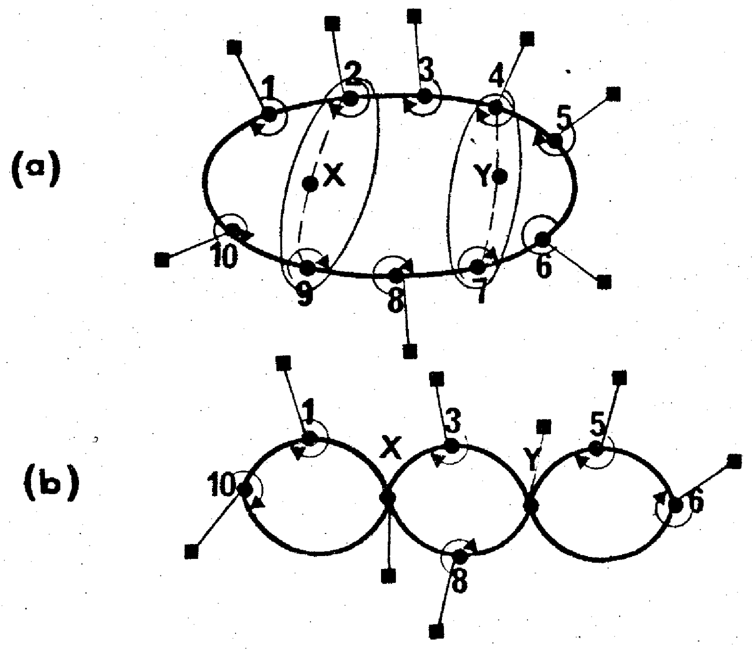


Fig. 3.4.1 (a) partially oriented graph, modelling the circular ordering of the terminals. (b) partially oriented graph modelling physical equivalence.

We can now define the following condition for the model to satisfy the conditions of section 3.4.1. For every set F(j) = {b(j,k)} of cardinality greater than 1, we add a K(1,n) subgraph to the model, with a new vertex d(j) being the center, connected to the vertices b(j,k) such that d(j) $><$ [c(j,k),a(j,k)], where b(j,k) = S(a(j,k)) and c(j,k) = S(b(j,k)). If the oriented graph so obtained is outerplanar, then the graph derived by contracting all the K(1,n) subgraphs is also outerplanar.

The above considerations are valid only when the original graph is oriented. When a mirror-image component is available, no proper model for physical equivalence was found. This is illustrated in Fig. 3.4.2, where the embeddings in (a) and (b) are correct, while the ones in (c) and (d) are not. Fig. 3.4.2 (e) shows the original oriented graph, modelling the circular relationship between the terminals.

3.4.4 Logical Equivalence of Terminals.

Let C = {L(i)}, where L(i) is a set of logically equivalent terminals. Then $|L(i)| \geq 1$ and $L(i) \cap L(j) = \emptyset$ if $i \neq j$. Let C' = {p(j)} be the set of terminals. Then the family $\tilde{L}$ of subsets L(i) of C' forms a partition of C'.

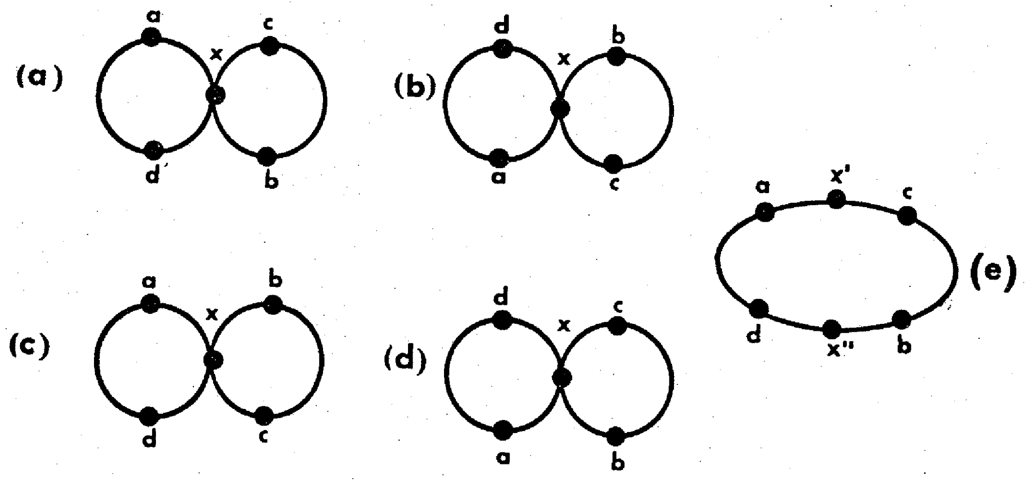


Fig. 3.4.2 Modelling of physical equivalence when a mirror image component is allowed.

Consider a set L(i) = {q(j)} of terminals with identical logical functions. Each of the terminals q(j) is incident with a net N(k). The component model should be outerplanar and for every permutation of the terminals, the cyclic ordering should be respected.

The cyclic ordering, specified by the successsor function, can be written as a relation R: p(1) R p(2), p(2) R p(3), ... p(n) R p(1). This successor function induces a relation R' on the elements of the family $\tilde{L}$: for all p(i), p(j) such that p(i) R p(j) with p(i) $\in$ L(k) and p(j) $\in$ L(l): L(k) R' L(l).

Let G be the partially oriented graph, representing the component boundary and its incident net-edges. We can then consider the partially oriented graph G', obtained by contracting the sets L(i). The vertices of G', corresponding to sets L(i) of cardinality 1 remain oriented. An example is shown in Fig. 3.4.3.

In order to perform pin assignment in function of the layout, the orientation of the nets around a vertex corresponding to a set L(i) of cardinality greater than 1 should not be specified. This can be accomplished by inserting an edge for every set L(i) of cardinality greater than 1, as shown in Fig. 3.4.3(c). Let G[L(i)] be the subgraphs of the directed cycle, generated by the subsets L(i). Then there exist two possibilities:

1) All of the subgraphs G[L(i)] are connected. This implies that the vertices of L(i) are physically adjacent. Then the new graph G' remains 2-connected. A vertex in G', corresponding to a set L(i) of

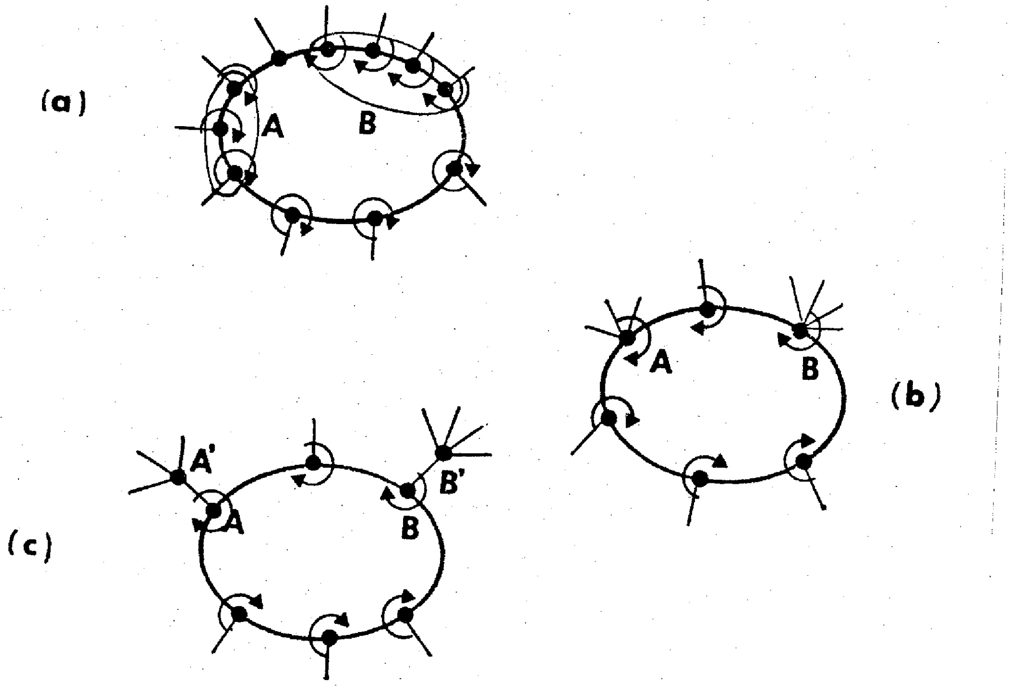


Fig. 3.4.3 (a) Circuit with logically equivalent sets A and B. (b) After contracting the sets A and B. (c) Correct oriented graph model.

cardinality n, will then be connected to n nets. Assuming that the total circuit layout graph is planar, then every possible embedding will satisfy the requirements of section 3.4.1.

2) At least one of the subgraphs $G[L(i)]$ is disconnected. This implies that the vertices of $L(i)$ are not all adjacent. As a result the graph G' is 1-connected and the set $L(i)$ is mapped into a separating vertex. If $L(i)$ has cardinality n, then the corresponding vertex will be connected to n nets. Not every possible embedding however satisfies the requirements of section 3.4.1.

Therefore, this model is appropriate for logical equivalence, if all logically equivalent terminals are physically adjacent. An interesting case, in which logical equivalence can be modelled even if the terminals are not physically adjacent, is illustrated in Fig. 3.4.4.

The condition for being able to model the logical equivalence of non-adjacent sets of cardinality 2 is similar to the one for physical equivalence, as pointed out in section 3.4.3. The model derived, however, is different. This shows that it may be possible to model logical equivalence even if the terminals are not physically adjacent.

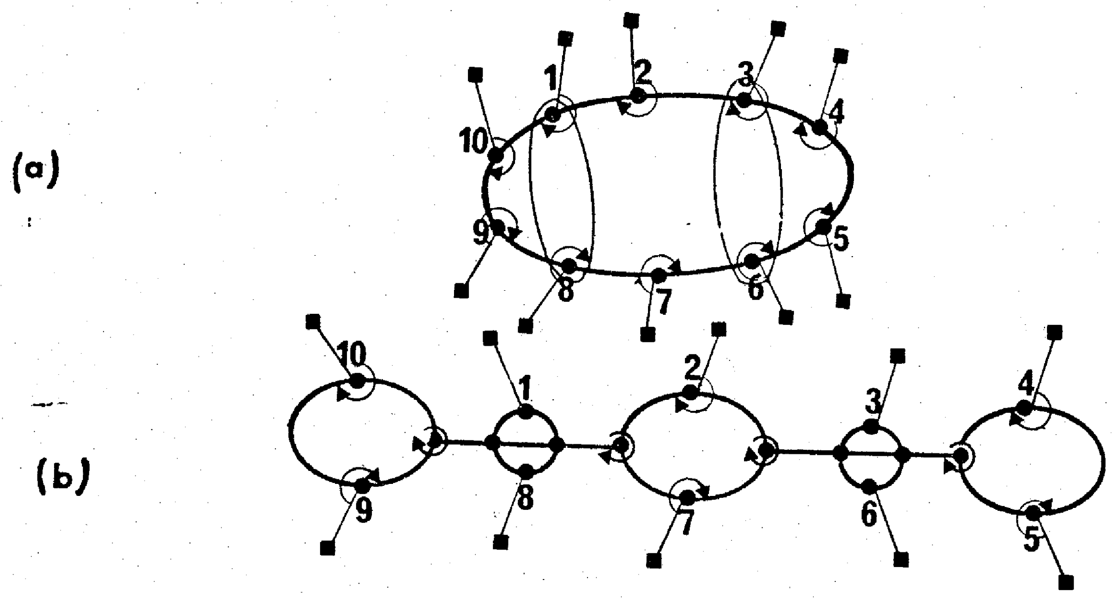


Fig. 3.4.4 Modelling of logical equivalence for sets of cardinality 2 that are not physically adjacent.

When a mirror-image component is available, a simple non-directed cycle models the order of the terminals on the component's boundary. Again, when the terminals in a logically equivalent set are physically adjacent, a correct model can be derived by contracting the set of edges. In order to avoid the embedding of connections inside the cycle, a star-subgraph should be placed inside the resulting cycle.

If it is necessary to represent every terminal by a distinct vertex, then a set L(i) of cardinality n can be replaced by a K(1,n) star. An example is shown in Fig. 3.4.3(c).

3.4.5 Unspecified Order of Terminals.

In some cases, the components can be (re-)designed if this would result in a better layout. From the point of view of circuit layout, this property is similar to logical equivalence of terminals.

In case the order of all terminals is unspecified, the component can be represented by a single undirected vertex (or by a K(1,n) star-subgraph if one desires a distinct vertex to represent each terminal). This results in a model, similar to the one proposed by Engl and Mlynski (EM73).

3.4.6 Logical Equivalence of Subcomponents.

Let C be a component, consisting of n sets E(i) of logically equivalent subcomponents. Let C' be the set of all subcomponents $S(j)=\{p(j,k)\}$. Then the family $\vec{E}=\{E(i)\}$ partitions the set C' into disjoint subsets.

In the following discussion, we will impose the following restrictions:

1) a subcomponent is a collection of <u>physically adjacent</u> terminals, performing a specified logical function.

2) In order for 2 subcomponents to belong to the same set E(i), they have to perform identical logical functions, have the same number of terminals and the order in which the terminals appear on the component's boundary must be the same.

Consider the partially oriented graph model G for the order of the terminals on the component's boundary, as illustrated in Fig. 3.4.5a and 3.4.6a. Let G[S(i)] denote the subgraph of G, generated by a subcomponent S(i). Since we require the terminals of a subcomponent to be physically adjacent, all G[S(i)] are connected paths. Let x(i) and

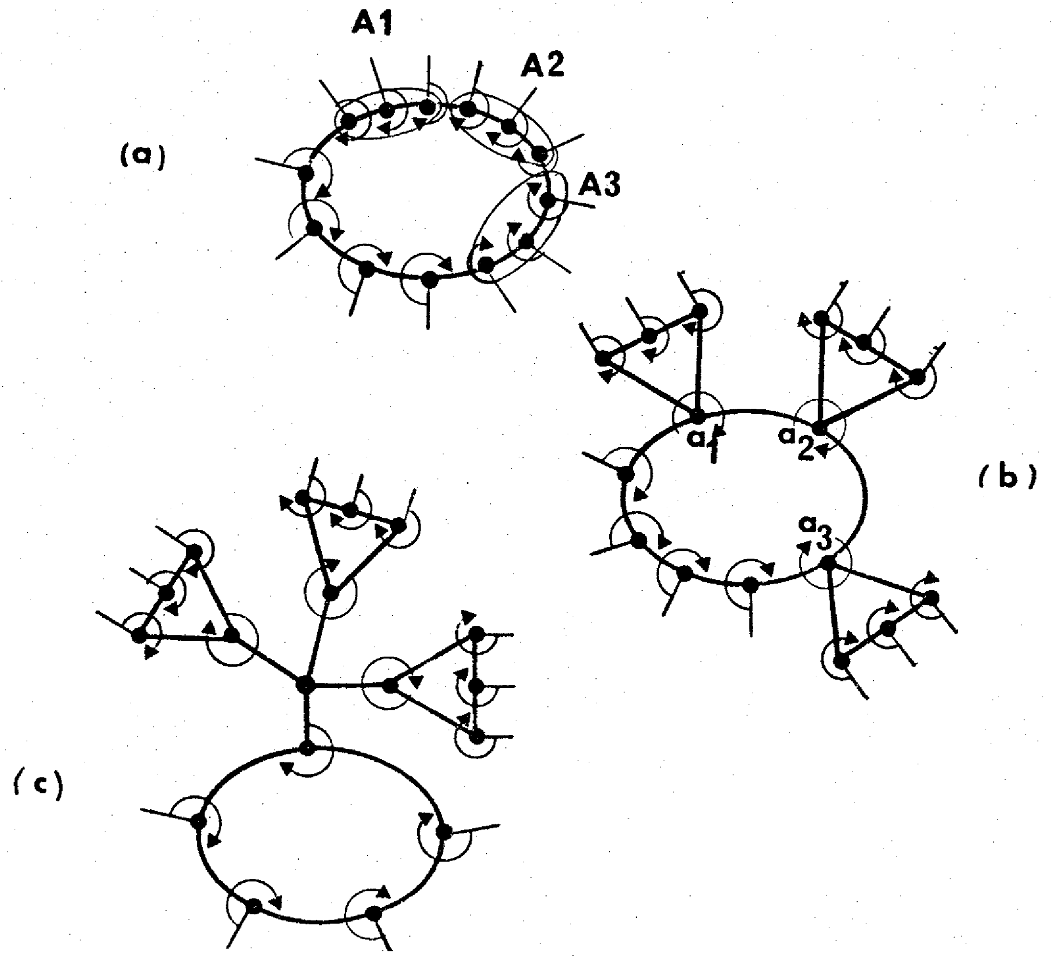


Fig. 3.4.5 Derivation of a model for subcomponent equivalence when equivalent subcomponents are physically adjacent.

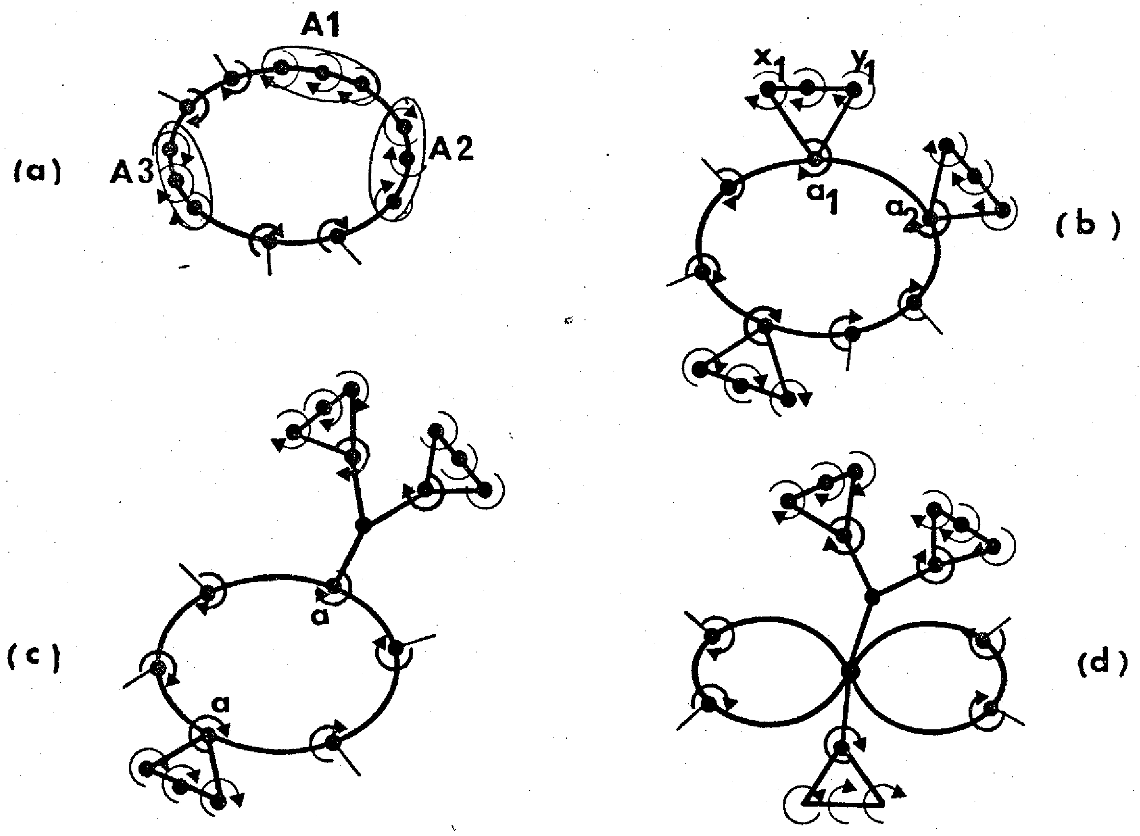


Fig. 3.4.6 Derivation of a model for subcomponent equivalence when equivalent subcomponents are not physically adjacent.

$y(i)$ be the vertices of valency 1 of $G[S(i)]$, such that $y(i) = S^+(x(i))$. † Let $x_1(i)$ be the vertex preceding $x(i)$ i.e. $x(i) = S(x_1(i))$ and let $y_1(i)$ be the vertex following $y(i)$ on the original cycle i.e. $y_1(i) = S(y(i))$. We now derive a new graph model G' for the component as follows.

$V(G') = V(G) \cup \{a(i)\}$

$E(G') = E(G) - \{(x_1(i),x(i)),(y(i),y_1(i))\}$
$\cup \{(x_1(i),a(i)), (a(i),y_1(i)), (a(i),x(i)), (a(i),y(i))\}$

where $a(i)$ is a new vertex, associated with subcomponent $S(i)$.

By repeating this for every subcomponent $S(i)$ with more than 1 terminal, we obtain a new oriented graph model G". For the purpose of embedding, G" is equivalent to the original model G. An example is given in Fig. 3.4.5b and 3.4.6b.

As a result of this transformation, the graph model now consists of an oriented cycle for the component itself and of oriented cycles for each of the subcomponents. The cycle modelling the component itself, contains vertices that represent terminals that did not belong to any

† By $a = S^+(b)$ we will denote that a was obtained from b by a multiple application of the successor function S.

subcomponent as well as vertices a(i) for each subcomponent.

Let E(i) = {S(j)} be a set of subcomponents with identical logical functions. Let every subcomponent S(j) be connected to a collection N(j) of nets. The component model to be derived should be outerplanar and for every permutation of the logically equivalent subcomponents, the cyclic ordering should be respected.

Consider the cycle for the component itself. This cyclic ordering of the vertices can be written as a relation a(1) R a(2), a(2) R a(3), ..., a(n) R a(1). Every subcomponent has exactly one vertex on this cycle. Hence, a set E(i) of logically equivalent subcomponents corresponds to a set E'(i) of vertices on the cycle. Then the relation R induces a relation R' on the subsets E'(i): for all a(i), a(j) such that a(i) R a(j), with a(i) $\in$ E(k) and a(j) $\in$ E(l): E(k) R' E(l).

We can now consider the oriented graph, obtained by contracting the edges (a(i),a(j)) of the cycle, such that both a(i) $\in$ E(k) and a(j) $\in$ E(k) for some k. Then two possibilities exist:

1) All vertices of a set E'(k) are physically adjacent (Fig. 3.4.5). When the total circuit layout graph is planar, then every possible embedding will satisfy the requirements of section 3.4.1.

2) The vertices of the set E'(k) are not physically adjacent (Fig. 3.4.6). Then not every embedding satisfies the requirements.

Therefore, we are able to model logical equivalence of subcomponents if the sets of terminals of all subcomponents that are equivalent are physically adjacent.

It should be pointed out here that physical and logical equivalence of terminals can be combined with subcomponent equivalence, as long as the model satisfies the conditions of section 3.4.1. Mirror-image components can no longer be modelled, except in simple cases.

3.5 Graph Models for the Boundary of a Circuit.

In section 2.4, the problem of modelling the outside connections of a circuit was described. When the order of these connections is predetermined, then we can again use a directed cycle. All nets now have to be embedded in the interior region (as opposed to the exterior region for components). This can also be modelled by an oriented graph.

In some cases, a certain degree of freedom exists, allowing logical signals to be assigned to physical terminals in function of the layout. This property is similar to the logical equivalence of the terminals of a component. As for components, the condition for being able to model the logical equivalence of a set of terminals is that they are physically adjacent. This then leads to the model proposed in section 2.4. Again, one can derive an oriented graph model for logical equivalence similar to the one for components.

3.6 Requirements for a Circuit Layout Graph Embedding Algorithm.

In view of the models, derived in the previous sections, the requirements for embedding circuit layout graphs in the plane are summarized.

1) The external connections should be embedded on a peripheral cycle.

2) For certain vertices, the edges incident to it should be embedded such that a clockwise sweep around the vertex encounters these edges in a prescribed order. The reason for this requirement is due to the models for components derived before.

3) When the circuit layout graph is non-planar (which is usually the case), a number of edges should be deleted from the graph such the that remaining graph is planar. Only edges representing nets can be removed. The remaining edges have to be embedded in a second or consecutive layer.

4) Only net-edges, connecting terminals on the same component may be embedded on the inside region of a component's boundary. This means that no connections

are allowed over or under the physical area of a component. This restriction is necessary if we want to embed the circuit layout graph or a maximally planar subgraph thereof in a single plane.

3.7 Properties of Circuit Layout Graphs.

Let G be a circuit layout graph, where the components and the exterior boundary are modelled by simple cycles and the nets by K(1,n) stars. Then $|V(G)| = |P| + |N|$ and $|E(G)| = 2\ |P|$, where P is the set of terminals (pins) and N is the set of nets. The number of vertices in G is obviously equal to the number of terminals plus the number of nets. There are two types of edges: <u>component edges</u> and <u>net edges</u>. Since components and outside connections are modelled by cycles, there are exactly as many component edges as there are terminals. Every terminal is connected to at most one net. Therefore, there are at most as many net edges as there are terminals.

We can consider the following extreme cases:

1) If $|N| = 1$ then $|V(G)| = |P| + 1$ and $|E(G)| = 2\ |P| = 2\ |V(G)| - 2$.

2) If $|N|$ is maximal, then every net is of cardinality 2 and $|N| = |P|/2$. Then $|V(G)| = 3/2\ |P|$ and $|E(G)| = 2\ |P| = 4/3\ |V(G)|$.

Thus $2\ |V(G)| - 2 \geq |E(G)| \geq 4/3\ |V(G)|$

The importance of this relation lies in the fact that the number of edges of G is a _linear function_ of the number of vertices. This is relevant when evaluating the computational complexity of algortihms that manipulate circuit layout graphs.

When the component models become more complicated, E(G) will still be a linear function of V(G) since the component models are always planar.

4. Embedding Circuit Layout Graphs.

4.1 Introduction.

In this chapter, the problem of finding a topological embedding of a circuit layout graph is studied. First the special requirements imposed on this embedding are given. It is shown that the problem of embedding a circuit layout graph on a multilayer surface cannot be solved by simply partitioning the graph into a minimum number of planar subgraphs and then embedding each of these subgraphs independently.

Since the actual layout problems are in practice very large, the computational complexity of the solution is very important. This is discussed in section 4.1.3.

Most of the existing planarity testing methods are based on trying to construct an embedding of the graph in the plane. Therefore, some of the more important algorithms are surveyed in section 4.2. Attempts to use these algorithms for solving the circuit layout problem are discussed.

In section 4.3, a new $O(|V|)$ algorithm is presented for testing the planarity of partially oriented graphs. This algorithm is then used in two approximate algorithms for finding a maximal planar subgraph of a circuit layout graph.

4.1.1 Requirements for Embedding Circuit Layout Graphs

In the previous chapter, it was shown that circuit layout graphs require an embedding algorithm with the following constraints:

- Certain vertices have to be positioned on the periphery.

- For certain vertices, there is a prescribed order, in which the edges incident to it must be embedded.

- If G is non-planar, a minimal number of edges has to be removed to make the remaining graph planar with the constraint that certain edges (specifically those representing component peripheries) cannot be removed.

In most cases, the graph G is non-planar and can therefore not be embedded in a single layer. The approach taken here is to embed a maximal planar subgraph in the first layer, subject to certain constraints, as pointed out in section 4.3. This approach is acceptable when the technological properties of the problem require most of the connections to be in the first layer (e.g. for integrated circuit layout).

Unless otherwise indicated, components will be modelled by connected subgraphs and n-terminal nets by K(1,n) star subgraphs.

The geometrical properties of the problem will be neglected in this phase. Only the topological aspects will be considered. The result of the embedding algorithm is a list of oriented faces of a maximal planar subgraph as well as a list of the net-edges that were deleted.

4.1.2 The Circuit Layout Graph Embedding Problem.

In all but very simple cases, the circuit layout graph is non-planar. Usually the aim of a circuit layout procedure is to embed the graph in a minimal number of layers. Depending on the technological properties of the problem, additional constraints may be imposed. E.g. in the layout of integrated circuits, one could try to embed as many connections as possible in the first layer, while the other layer(s) are used only for embedding the remaining edges. In printed circuit layout, it would be preferred to have layers containing approximately the same number of connections, since this would probably make a better use of the area available.

In this chapter, these additional constraints, due to technological factors, will not be considered. The procedure proposed, however, is based on embedding a maximal planar subgraph in the first layer and therefore it is most suited for IC layout.

An important characteristic of embedding circuit layout graphs is that on <u>all layers</u> the components have the <u>same relative positions</u> with respect to each other. In other words, the embedding of a planar subgraph in one layer influences the embedding in the other layers. Furthermore, it is necessary to represent the components by appropriate graph models _on every layer_ in order to obtain a meaningful result. This differentiates the circuit layout problem from partitioning a graph into a minimum number of planar subgraphs. We will illustrate this difference by means of an example, shown in Fig. 4.1.1, where 3 components A, B and C are connected to each other by 6 nets N_1, N_2, ..., N_6.

For integrated circuit layout, the area used by a component may not be available for routing connections in any other layer. Therefore, the edges, modelling the component boundary have to be present in every layer.

Fig. 4.1.2 shows a possible decomposition of the graph into a minimum number (i.e. 2) of planar subgraphs.

If we adhere to the restrictions, imposed on the embedding of circuit layout graphs, then we have to embed every component edge on every layer. A minimal embedding, requiring three layers, is shown in Fig. 4.1.3.

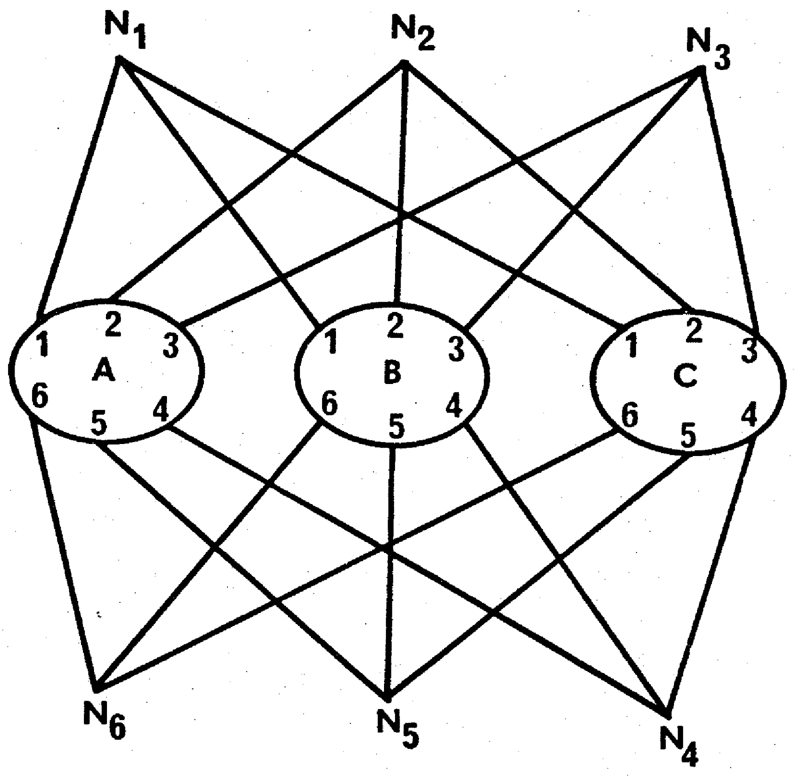


Fig. 4.1.1 Non-planar circuit layout graph.

Several authors (e.g. Ch70) have suggested a simple decomposition into planar subgraphs as a solution to the circuit layout problem. Even if the graph model used is a simple bipartite graph, as used by Chein (Ch70), with components represented by single vertices, then this partitioning into planar subgraphs does not necessarily result in a solution that is adequate for circuit layout.

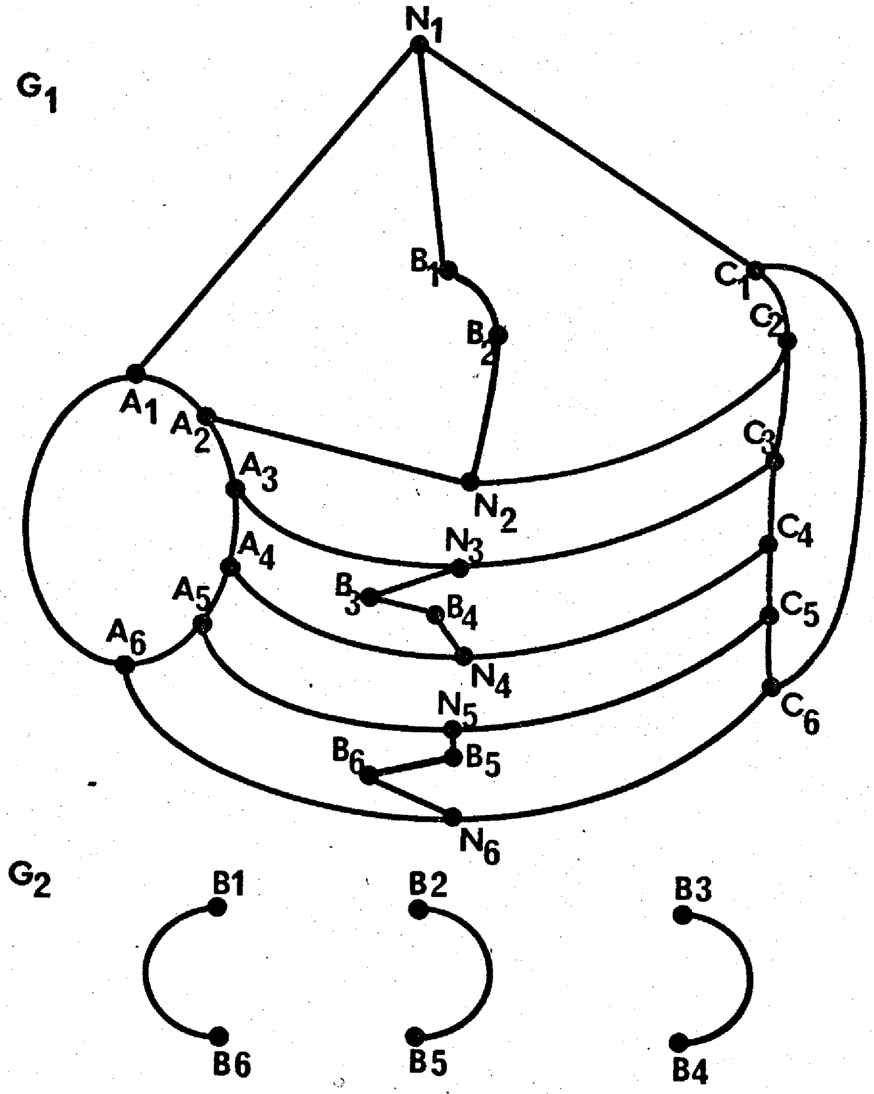


Fig. 4.1.2 Partitioning of the circuit layout graph into two planar subgraphs.

The approach taken here is to find a maximal planar subgraph first and to embed it in the first layer. The

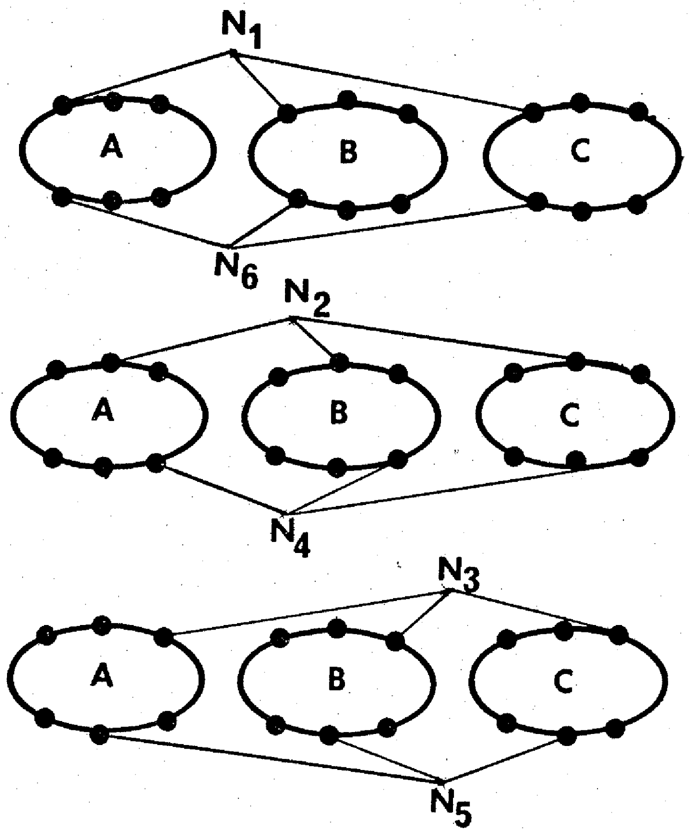


Fig. 4.1.3 Embedding the circuit layout graph of Fig. 4.1.1.

connections on the other layers can then be embedded using a classical routing algorithm (e.g. Hi69).

4.1.3 Computational Complexity of the Maximum Planar Subgraph Problem.

It was shown earlier (in section 3.8) that, for a circuit layout graph where components are modelled by cycles and nets by stars, $|E(G)| = k.|V(G)|$ with $2 \geq k \geq 4/3$. Even if more complicated component models are used, k will be constant with respect to $|V(G)|$.

The complexity of finding a maximum planar subgraph of a non-planar graph, by deleting a minimum number of edges is still an open problem in computational complexity. † Currently, no algortihm is known that solves the problem in polynomial time. Attempts to reduce a known NP-complete problem (Ka72, GS74) to this problem have not yet produced an answer to the question of NP-completeness.

Since no efficient algorithm for finding a maximum planar subgraph is known, we will attempt to find a maximal solution by using an approximate, polynomial-time algorithm.

When solving circuit layout problems, the graphs being investigated can easily have 1,000 or more vertices and in

† M. Garey and D. Johnson, private communication.

such cases the computational complexity of the algorithm is extremely important.

4.1.4 Definitions.

A number of definitions related to graphs and their planarity was given in section 3.0. We will introduce some additional terminology here.

The following definitions, related to efficient algorithms are from (HT74).

The _adjacency list_ of a vertex v is an unordered list of the vertices, adjacent to v.

The _adjacency structure_ A of a graph G is the collection of adjacency lists for all vertices of G.

In a directed graph, v is an _ancestor_ of w and w is a _descendant_ of v if there exists a path from v to w. If the path is of length 1, then v is the _father_ of w and w is the _son_ of v.

In a circuit layout graph, there exist two types of vertices: _C-vertices_ belonging to a component model and

N-vertices, being the center vertices of K(1,n) subgraphs modelling n-terminal nets. Similarly, the edges can be classified as C-edges, incident to two C-vertices and N-edges, incident to one C-vertex and to one N-vertex.

4.2 A Survey of Graph Embedding Algorithms.

In this section, a number of constructive planarity testing algorithms are discussed. These algorithms attempt to construct a planar embedding of the graph in the plane and as such are of interest in solving the circuit layout problem. Where possible, attempts to use the algorithm for circuit layout are discussed. It is assumed here that graphs are 2-connected without loops or multiple edges. If a graph is not 2-connected, it can be split into 2-connected subgraphs, and each of these can be treated separately. A graph is non-planar if $|E| > 3|V| - 6$. Therefore, only the case where $|E| \leq 3|V| - 6$ has to be investigated.

4.2.1 Fisher and Wing's Algorithm.

This algorithm has been used in several topological layout systems (e.g. EM73, Kl73, BR74). It will be described using Read's formulation (Re70).

1) Find a cycle C.

2) Delete all vertices of C from G. The resulting graph consists of a number of connected components F(i), called fragments. In G, a fragment F(i) is connected to C by a set L(i) of link-edges. There also might be edges t(j) in G, joining two vertices of C. They will be called transversals. Without loss of generality, the transversals can be considered as fragments by placing a vertex in the middle of each transversal.

3) Each fragment F(i) has to be embedded either on the inside or on the outside of C. Consider the subgraphs H(i) of G, formed by C, F(i) and L(i). If all H(i) are planar, then G is planar if all of the fragments F(i) can be embedded such that their link-edges do not cross.

Let two subgraphs G_1 and G_2 be called compatible if both can be embedded on the same side (inside or outside region) of C, without a crossing. Define a compatibility graph K as follows. For every fragment F(i), there is a vertex in K. Two vertices of K are joined by an edge if and only if the corresponding fragments are incompatible. Then the original graph is non-planar if there does not exist a 2-coloring of the vertices of K.

4) If the fragments can be embedded without crossings, then the problem is reduced to testing the planarity of each of the subgraphs H(i).

In (EM75), an iterative algorithm for testing the planarity of a multiplace graph is given. In the first step, the graph is decomposed into a pseudohamiltonian cycle H, a set G_{Δ} of transversals and a collection of subgraphs G_{σ}, that remain after deleting the cycle H and the transversals. The graph is then transformed into a multiplace graph by replacing the subgraphs G_{σ} by spiders. † In his description of Fisher and Wing's algorithm, Read (Re70) points out that the subgraphs G_{σ} can be replaced by star subgraphs, which is exactly what is proposed in (EM75).

A complicated method is developed to determine whether the subgraphs G_{σ} and the set of transversals G_{Δ} can be embedded without crossings. Theorem 4 in (EM75) is similar

---

† For a definition of spiders and multiplace graphs, see section 3.2.

to Read's formulation of Fisher and Wing's algorithm (Re70, FW64, FW66).

According to Shirey (Sh69), the runtime of Fisher and Wing's implementation has a lower bound of at least $O(|V|^4)$ whereas Tarjan's algorithm (Ta71) has a lower bound of $O(|V|)$, where $|V|$ is the number of vertices in the graph.

It is not easy to obtain a list of faces when using this algorithm to embed a planar graph. In fact, several implementations of this algorithm for solving the circuit layout problem make use of interactive graphics to obtain a plane drawing of the graph.

Some of the restrictions, imposed on circuit layout graphs, can be taken into account quite easily. E.g. the peripheral cycle can be enforced to be the external face of the embedded graph by selecting it as the first cycle and by requiring all fragments to be embedded on the inside with respect to it.

Enforcing the orientation of the vertices adjacent to a given vertex, is rather difficult.

4.2.2 Demoucron, Malgrange and Pertuiset's Algorithm.

The algorithm, described in (DM64) is similar in nature to Fisher and Wing's algorithm. In fact, both algorithms are variations of Goldstein's algorithm (Go63).

Rubin (Ru75) describes an implementation of this algorithm, requiring $O(|V|^2)$ time and space (no proof of this bound is given). Although it is claimed in this paper that for 'practical' graphs, the algorithm tends to behave linearly, no sufficient evidence is available to substantiate this claim.

An important characteristic of this algorithm is that it generates a list of the faces of the embedding, as required for solving the circuit layout problem.

4.2.3 Lempel, Even and Cederbaum's Algorithm.

The approach, used in this algorithm (LE66), is quite different from the previous ones. From the graph, $|V|-1$ directed rooted trees † are constructed. Each tree consists of a root vertex and of the outdirected edges, incident to the root. The planarity testing method is based on constructing G in the plane from the rooted trees.

Tarjan(Ta71) reports an implementation of this algorithm in $O(|V|^2)$ time.

Using this algorithm, it is easy to find a list of the faces in the embedding. Furthermore, if G is non-planar, deleting an edge to keep the partially constructed graph planar is easily implemented. Although this will create a planar subgraph of the original graph, this subgraph is not necessarily maximum. However, if the component models are complicated (as proposed in the previous chapter), then it might become necessary to delete a C-edge, contrary to the requirements of section 4.1.

† A directed rooted tree T is a directed graph with a special vertex r, called the root, such that every vertex in T is reachable from r, no edges enter r, and exactly one edge enters every other vertex.

Uyehara et al. (US74) report a modification to this algorithm that takes into account the prescribed orientation of edges around certain vertices, as is required for testing the planarity of circuit layout graphs.

In order to satisfy the requirements, outlined in section 4.1, it would be necessary to first embed all the C-edges and then add the N-edges, one at a time, each time testing the new graph for planarity as described in (US74). This requires $O(|V|^3)$ time if $|E| = O(|V|)$.

4.2.4 Tarjan's Algorithm.

Tarjan's algorithm (Ta71, HT74) forms the basis for the algorithms, described in section 4.3. It is required that the graph being tested is 2-connected. The graph is specified in the form of an adjacency structure A. The algorithm consists of the following major steps:

1) apply a depth-first search to G, starting from some (arbitrarily chosen) vertex s. This search imposes a direction upon the edges of G, depending on the order

in which its end-vertices were reached by the search. By doing so, G is transformed into a directed graph G', whose edges are partitioned into a set of edges, forming a spanning tree, and a set of _fronds_. The directed graph G' is called a _palm tree_.

2) In this step, the adjacency structure A is ordered, using the information obtained in step 1. This ordering is done using a radix sort in time O(|V|) (HT74). The reason for this ordering is to be able to perform step 3 efficiently.

3) Partition the graph G into a set of edge-disjoint paths, by performing a depth-first search on the adjacency structure, ordered in step 2. Each time an edge is traversed, it is added to the current path. When the edge is a frond, the current path ends and a new path is started.

This procedure requires O(|V|+|E|) time (HT74). Because of the ordering of the adjacency structure, these paths have special properties, which will allow efficient testing of the planarity (Lemma 5 to 8 in (HT74)). Let p = (s,...,f) be a path and let p_0 =

$(s_0, \ldots, f_0)$ be the first path generated containing s. Then p is a _special path_ if $f = f_0$ and a _normal path_ otherwise (Ta71).

4) Embed the first path (a cycle) in the plane as a polygon. Try to embed the other paths, in the order in which they were generated in step 3), one at a time. Each new path has exactly two points in common with the already embedded subgraph. Certain paths have to be embedded in different faces or in the same face, with respect to other paths. This relationship between paths can be represented by a _dependency graph_. Tarjan proves that it is sufficient to construct only a subgraph of this dependency graph, for which the number of edges is a linear function of $|V(G)|$.

5) Try to bicolor the dependency subgraph, using the colors L (left) and R (right). A path $P = (s,u,..,f)$ is said to be _embedded on the left (right)_ if (s,u) is the first edge clockwise (counterclockwise) from (v,s) in the list of edges incident to s, where (v,s) is the edge of the spanning tree entering s. A frond (w,f) _descends on the left (right)_ if it is the first

edge clockwise (counterclockwise) from (x,f) in the list of edges, incident to f, where (x,f) is the edge of the spanning tree entering f. If this is not possible then G is non-planar.

Tarjan's algorithm for testing the planarity of a graph is linear in $|V(G)|$. If the graph is not 2-connected, then it can be decomposed into 2-connected components in $O(|V|)$ time and the planarity of each of these 2-connected components has to be tested separately.

4.2.5 Brehaut's Face-finding Algorithm.

Although Tarjan's algorithm tests the planarity of a graph by embedding it in the plane, it is not obvious from (HT74, Ta71) how one could obtain a list of the faces of the plane graph. In (Br74, Br75), an efficient algorithm is given for this problem. It is an extension of Tarjan's algorithm, requiring $O(|V|)$ time and space.

First, a path tree is constructed, in which a path P_1 is a son of a path P_2 if and only if P_1 debuts on P_2. The adjacency structure, representing the path tree, is then reordered in order to allow an efficient search for the faces. By traversing the new adjacency structure, using a depth-first search, the paths are encounterd in the correct order. Each time a path is added to the embedding, one of the existing faces is split into two new faces.

4.3 Embedding a Maximal Planar Circuit Graph.

4.3.1 Planarity Testing for Partially Oriented Graphs.

This section is concerned with an extension to Tarjan's algorithm for testing the planarity of a partially oriented graph G in linear time. We can reduce the problem to testing the planarity of partially oriented graphs with oriented vertices of valency 3 only. This can be accomplished by replacing every oriented vertex of valency n, greater than 3, by a cycle with n new oriented vertices of valency 3, as illustrated in Fig. 4.3.1.

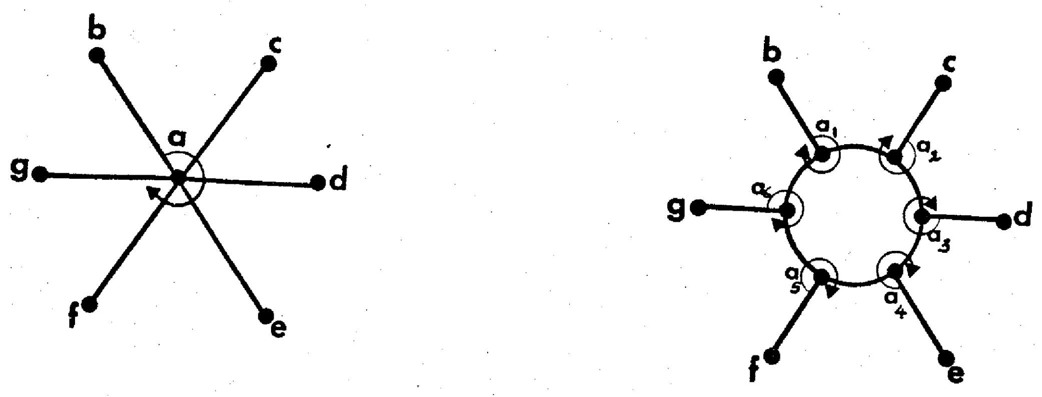


Fig. 4.3.1 Replacing an oriented vertex of valency n by oriented vertices of valency 3.

Let G' be the partially oriented graph, obtained by this transformation. Then the following holds.

<u>Theorem</u>: G' is planar if and only if G is planar.

<u>Proof</u>: a) suppose G' is planar. Then G can be obtained from G' by contracting the cycles that replaced the oriented vertices of valency greater than 3. Then G is planar.

b) If G is planar, then G' is planar. If G is planar, then there exists a cycle basis Z(1),...Z(m) and a cycle Z(0) in G, such that every edge of G belongs to exactly two of these cycles (MacLane). Then, because of the transformation defined, there exists a cycle basis Z'(1),...,Z'(n) and a cycle Z'(0) in G', such that every edge of G' belongs to exactly two of these cycles.

For most circuit layout graphs, oriented vertices will be of valency 3. If not, it will be assumed that the component models have been transformed such that this property holds.

It can easily be verified that $|E(G')| \leq 3|E(G)|$ and that $|V(G')| \leq 2.|E(G)|$. Therefore, if $|E(G)| = k.|V(G)|$ then $|E(G')| = k'.|V(G')|$.

The following algorithm is a modification of Tarjan's algorithm. It allows one to test the planarity of a partially oriented graph G(V,E) with oriented vertices of valency 3 only, in time proportional to $|V|$ if $|E| = k.|V|$.

1-3) Steps 1-3 are essentially the same as in Tarjan's algorithm (section 4.2.4).

4) Examine all paths p = (s,...,f). If s is an oriented vertex, then determine on which side (Left or Right) p has to be embedded in order to satisfy the orientation imposed around s. If f is oriented, then determine on which side the frond has to descend. We can represent this as a function SIDE(p,v) with possible values L(eft), R(ight) and U(ndefined). SIDE(p,v) = U if vertex v is not oriented.

First assume that p is a normal path. Then the frond of p has to descend on the same side as the embedding of the path. The following decision table

SIDE(p,f)	SIDE(p,s): L	R	U
L	l	n	l
R	n	r	r
U	l	r	u

where l: embed p on the left.
r: embed p on the right.
u: p can be embedded on both sides.
n: the graph is non planar.

indicates the action to be taken for each possible case.

If p = (s,...,f) is a special path, then its frond has to descend on the same side as the frond of p_0, i.e. the path on which s debuts, unless $f=f_0=1$. However, if $f\neq 1$ then the valency of f must be at least 4 and therefore f cannot be an oriented vertex. If f=1 and oriented, then we can require without loss of generality that the frond of p descend on the same side as the path's embedding (similar to normal paths). The only case left for special paths is when s is an oriented vertex. Then the embedding of p is determined by SIDE(p,s).

5) Build a dependency graph. † This step is esentially the same as in Tarjan's algorithm (section 4.2.4, step 4).

6) Try to bicolor the dependency graph DG, using the colors L(eft) and R(ight). DG consists of one or more disconnected components. The difference with Tarjan's algorithm is that there are a number of vertices in V(DG) that have preassigned colors. Every vertex in V(DG) represents a path in the original graph. The bicoloring procedure consists of the following major steps.

- mark all vertices as unexplored.

- find a colored, but not explored vertex and mark it as explored. If no such vertex exists, then find an unexplored vertex and assign it any color (e.g. L). If all vertices have been explored, then DG can be colored with 2 colors and hence G is planar.

- use a depth-first search to explore the component of DG, containing the selected start-vertex: each time a vertex is reached, check whether it is colored or not. If not, assign it the appropriate color. If it was already colored, check the colors: if they are compatible, continue; if not, the graph G is non-planar.

The procedure for enforcing the embedding of paths (step 4) requires O(|E|) time for general graphs and O(|V|) time if |E| = k.|V|. Bicoloring the dependency graph requires time proportional to the number of edges in E(DG). If a dependency subgraph was constructed and if |E| = k.|V| then |E(DG)| = O(|V|) and the time required for testing the planarity of G is O(|V|).

4.3.2 Finding a Maximal Planar Circuit Layout Graph 1.

As already pointed out in section 4.1, the computational complexity of finding a maximum planar subgraph of a non-planar graph is unknown. The same is true for oriented graphs. Therefore, we will be satisfied with a maximal solution.

One possible algorithm is as follows. Apply the algorithm of section 4.3.1 up to step 5 (i.e. finding a dependency graph). Then try to 2-color DG, until an incompatible vertex v is reached. Delete this vertex. Deleting a vertex v of DG corresponds to deleting a path p in the original graph. In fact, it is sufficient to delete a single edge of this path. However, the dependency graph is no longer a valid representation of the resulting graph. Therefore, as soon as an edge of the original graph has been deleted, steps 1-5 of the planarity testing algorithm have to be repeated in order to construct a new dependency graph. This has to be repeated until the dependency graph can be 2-colored and hence the resulting graph is planar.

It should be noted that the planar subgraph obtained in this way is not necessarily maximal.

When circuit layout graphs are considered, it is not permitted to delete C-edges. Only N-edges may be deleted. Therefore, the paths, found in the pathfinding procedure can be classified as C-paths (containing C-edges only) and N-paths (containing at least one N-edge).

The following approach can be taken for 2-coloring the dependency graph such that no C-edge will be deleted.

- construct and bicolor the subgraph of DG, generated by the vertices, corresponding to C-paths. This is always possible, since component models are planar and connected to each other by N-edges only.

- try to add the vertices, corresponding to N-paths, to the subgraph, as long as the 2-coloring can be maintained.

- if an N-path cannot be added, the delete an N-edge belonging to the path. Then reconstruct all paths and build a new dependency graph.

The time required for constructing a maximal subgraph of the dependency graph is $O(|E(DG)|)$ and the total time required for finding a planar subgraph is $O(|V(G)|^2)$ if $|E(G)|=k.|V(G)|$.

In order to obtain an optimal solution, a final step is required whereby the unembedded edges are added to the embedding where this is possible.

The algorithm, presented in the next section allows this to be done in $O(|V(G)|^2)$ time.

4.3.3 Finding a Maximal Planar Circuit Layout Graph 2.

The algorithm, presented in this section will always produce a maximal solution. Let G be a circuit layout graph, with vertex set V(G) and edge set E(G). Then E(G) = C U N, where C is the set of all C-edges and N is the set of all N-edges.

Rather than starting with the given circuit layout graph G as the solution and then deleting edges one at a time until the resulting subgraph is planar, as in the previous section, this algorithm starts with a planar subgraph of G (e.g. the null graph) and attempts to add edges, one at a time. The partially constructed graph will be referred to as H(V(H),E(H)).

1) instead of starting with the null graph, we can make use of the following property. Let G be a circuit layout graph. The graph, obtained by deleting all N-edges from G, consists of a collection of mutually disconnected subgraphs. Each of these subgraphs represents a component model and is planar. Then the graph, left after deleting all N-edges from G is

planar. Therefore let $H = (V,C)$ initially.

2) Some of the vertices in $V(H)$ will be of valency 0. Since an edge, incident to a vertex of valency 1 does not influence planarity, add edges of $E(G) - E(H)$ to H if at least one of their vertices has valency 0 in H. Assuming that the circuit has n nets and p connected terminals, then there are exactly p N-edges and n N-vertices. After step 1, there will be exactly n vertices in H with valency 0. This allows us to add exactly n N-edges to H, leaving $p - n$ edges still to be embedded.

3) At this stage, adding an edge to H might create a non-planar graph. Therefore, the following steps have to be carried out for every edge in $E(G) - E(H)$.

4) Select an edge of $E(G) - E(H)$ and add it to H.

5) Find the biconnected components of H.

6) For every non-trivial biconnected component, perform the planarity testing algorithm, described in section 4.3.1.

7) If at least one of the biconnected components is non-planar, then delete the edge from H (and G). This edge cannot be added later, thereby assuring that the solution is maximal (but not maximum).

8) While E(G) - E(H) $\neq \emptyset$, repeat steps 4-7.

9) Apply Brehaut's mesh-finding algorithm to the resulting graph in order to obtain a list of faces of the embedded graph.

An example of applying this algorithm to a circuit layout graph is given below. Consider the simple circuit of Fig. 4.3.2, implementing a full adder, using NAND gates.

We will attempt to embed this circuit under two different assumptions.

1) All NAND gates are to be considered as individual components. The graph model for this case is shown in Fig. 4.3.3.

   It is assumed in the model that the order of the external connections is determined in advance and that for every gate, a mirror-image equivalent is available. In the resulting maximal planar subgraph,

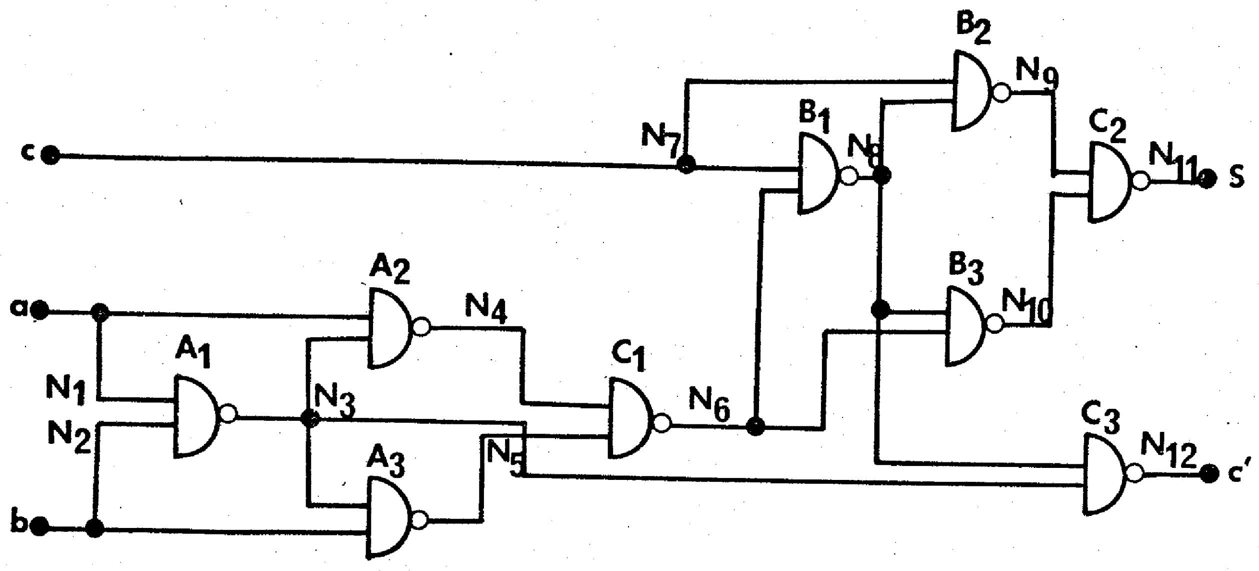


Fig. 4.3.2 Full adder implemented using NAND gates.

two net-edges were removed, as shown in Fig. 4.3.4.

The graph model contained 26 vertices and 37 edges and the time required was 1.2 seconds on an IBM 360/75.

2) The gates are to be grouped together into 3 components, each containing 3 gates. This grouping was done arbitrarily. No mirror-image components are allowed and the order of the external connections is

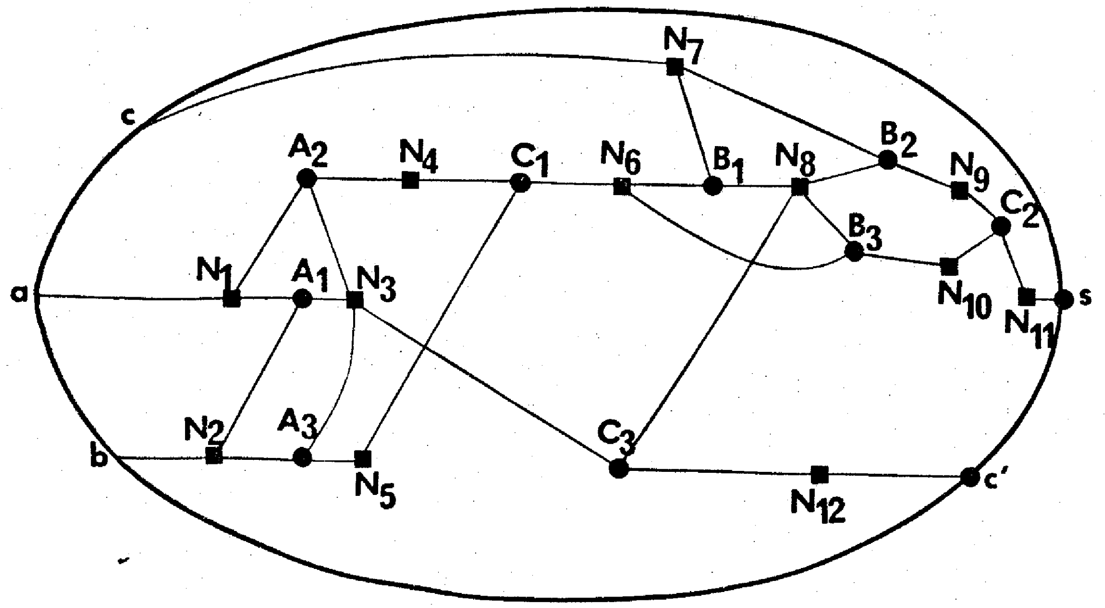


Fig. 4.3.3 Graph model for the full adder of Fig. 4.3.2.

again specified in advance. The graph model is shown in Fig. 4.3.5.

It contains 74 vertices and 100 edges. The resulting maximal planar subgraph is shown in Fig. 4.3.6.

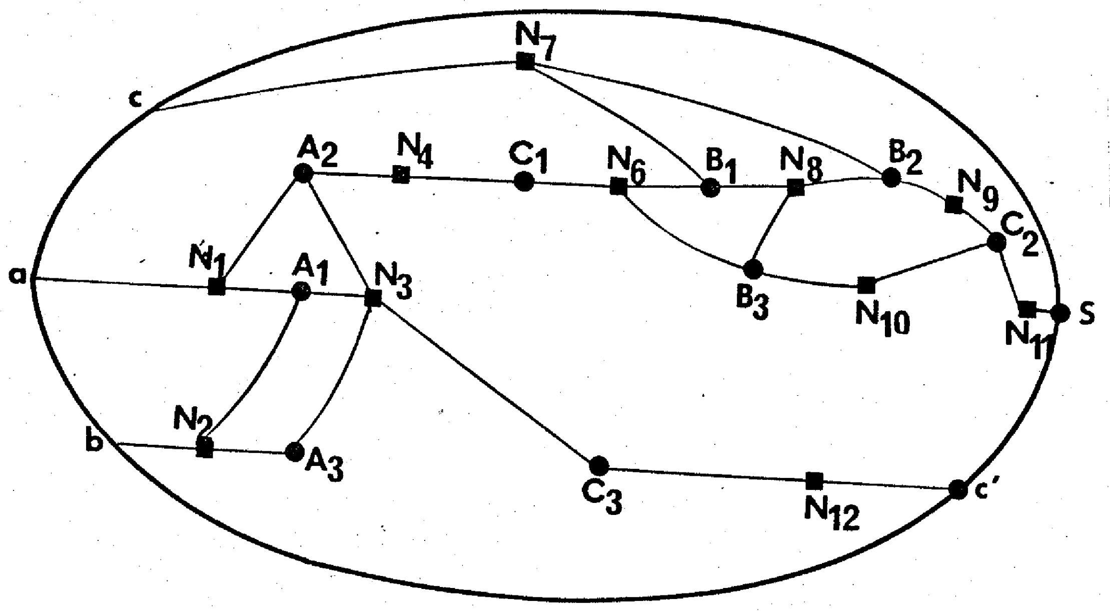


Fig. 4.3.4 Maximal planar subgraph of the model of Fig. 4.3.3.

Six net-edges had to be deleted. The execution time required to find the planar embedding was 6.5 seconds. Note that the layout procedure performed some degree of gate assignment. As an example, consider the subcomponents C1, C2 and C3. In the original graph model, they appeared in the sequence C1, C2, C3. In the embedded planar subgraph of Fig. 4.3.6 they appear in the sequence C1, C3, C2.

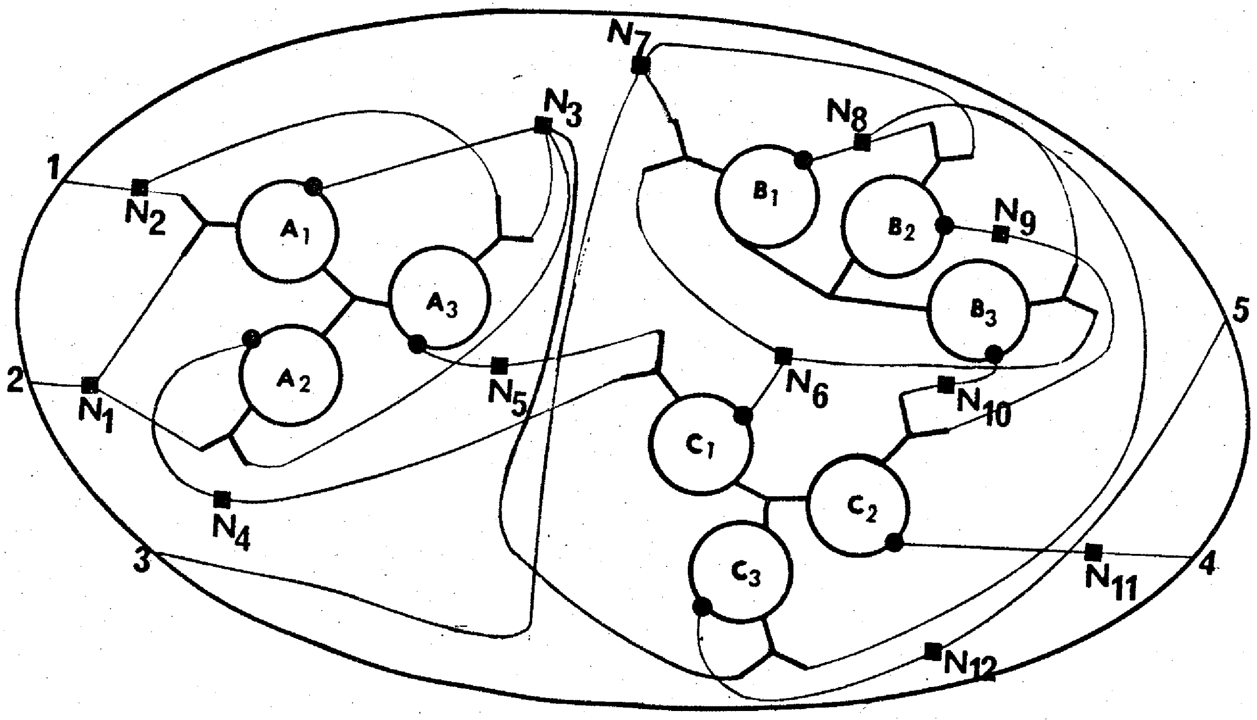


Fig. 4.3.5 Graph model for the circuit of Fig. 4.3.2.

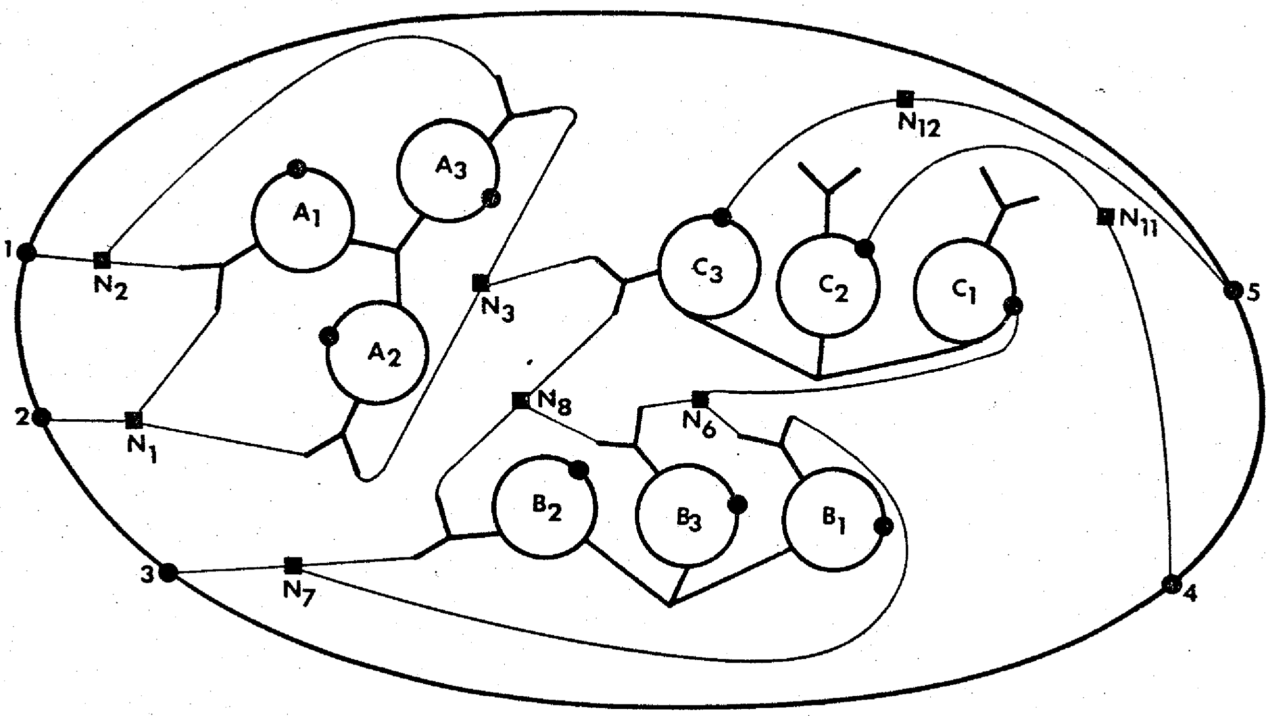


Fig. 4.3.6 Maximal planar subgraph of the model of Fig. 4.3.5.

5. Using the Technological Properties to Complete the Topological Layout.

In this chapter, some techniques will be discussed for using the properties that depend on the technological aspects of the problem. Most of these techniques are extremely dependent both on the particular layout problem being solved and on the industrial environment in which they are used.

Since it would be impossible to cover all possible situations, that could occur in an industrial environment, only some of the more common technological properties will considered and techniques for using these in a topological layout procedure will be suggested.

Some of these properties were already taken into account in the models, developed in chapter 3. For example, by using appropriate models, it is frequently possible to defer pin and gate assignment until a maximal planar subgraph is embedded. After this embedding, pin and gate assignments have to be done, according to the embedding that was obtained. The same holds for the occurrence of mirror-image components. Again, this property can be modelled by the circuit layout graph and, after embedding a maximal planar subgraph, it has to be determined whether the component itself or its mirror-image has to be used.

Sometimes it is possible to route connections between the terminals of a component and over or under the physical area, occupied by the component. This property was not modelled by the circuit layout graph. However, it can be used to embed connections, that were deleted when embedding a maximal planar subgraph.

Finally, once the topological layout is completed, the geometrical characteristics of the circuit have to be taken into account. An algorithm for constructing a preliminary physical layout from the topological layout is described in section 5.5.

5.1 Detecting Mirror-image Components.

Sometimes a mirror-image component is readily available. This is frequently true for integrated circuit layout. When this is the case, the component models, derived in chapter 3, will contain an undirected cycle that can be embedded in two different ways.

In order to avoid embedding connections inside the component's boundary, it is necessary to place a star subgraph inside the cycle (cfr. chapter 3). Therefore, the first step is to delete this star from the embedding. This will result in the cycle becoming a face. From the orientation of this face, it can be determined whether the original component or its mirror-image is to be used.

5.2 Performing Pin and Gate Assignment.

At this stage, the circuit layout graph (or a maximal planar subgraph thereof) has been embedded in the plane. In this embedding, components are represented by sometimes complicated graph models, while some of the net-edges might have been deleted.

In the further transformations of the circuit layout graph, these complicated component models are no longer needed. Therefore, it is desirable to replace these component models by simple cycles. The models, proposed in chapter 3, take into account physical and logical equivalence properties of terminals. In the topological embedding phase, these properties were used for obtaining a maximal planar subgraph.

As an example of this, consider the full adder circuit of Fig. 4.3.2. Assume that three NAND gates were placed in a single physical component. The graph model for this circuit is shown in Fig. 4.3.5 and a maximal planar subgraph is shown in Fig. 4.3.6. In the original circuit graph, the subcomponents of C appear in the order C1, C2, C3. In Fig. 4.3.6, these subcomponents were embedded in the order C1, C3, C2, thereby effectively performing gate assignment.

After the topological embedding phase, the complicated component models of Fig. 4.3.6 are no longer required and they can be replaced by simple cycles, as shown in Fig. 5.2.1.

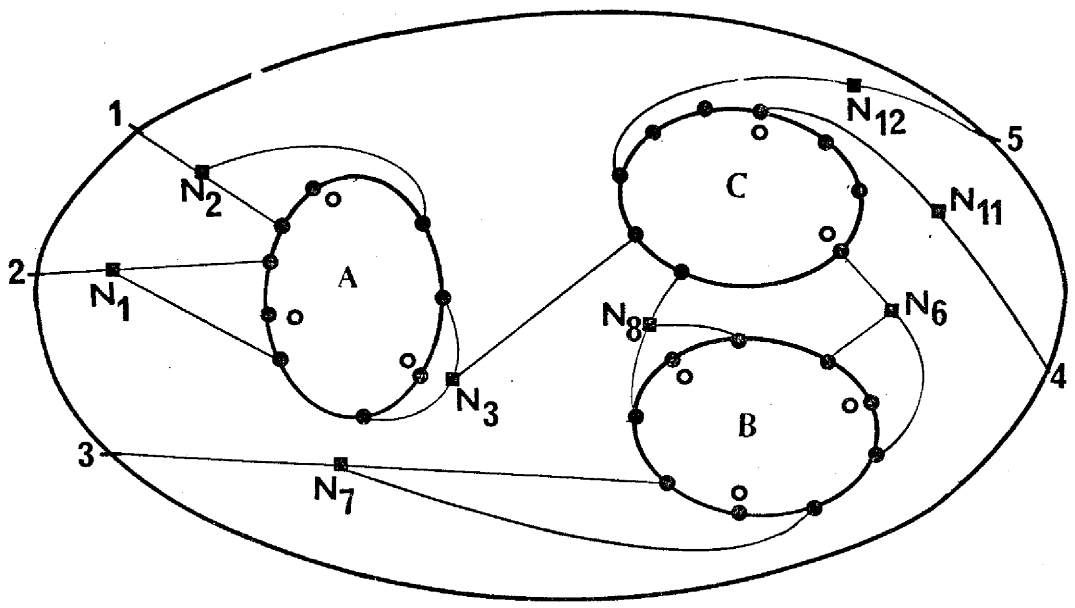


Fig. 5.2.1 Simplified graph model obtained from the maximal planar circuit graph of Fig. 4.3.6.

5.3 Using Finite Wiring Capacities.

The models described in chapter 3 do not allow wires to be routed between pins or under modules. Embedding of the graph will lead to a number of unrouted connections. A special characteristic of allowing wires e.g. between 2 adjacent pins is that these places have a finite wiring capacity. E.g. depending on the technology, one is allowed to route 0, 1, 2 or 3 wires between 2 adjacent pins of a dual-in-line package.

Fletcher (F172) proposed a solution whereby the edges that can be crossed by connections are assigned weights and whenever a 'pseudo crossing' is introduced this weight is decreased for the edge crossed.

In the method proposed here, finite capacities are represented by vertices; i.e if n wires are allowed between 2 adjacent pins, n new vertices will be placed on the edge connecting the 2 pins. It will be assumed here that final pin and gate assignments have been performed and that the graph has been embedded in the plane as far as possible. In this case the component can simply be represented by a cycle to model the sequence of its terminals.

As an example, consider a 14 pin DIP, with one wire allowed between 2 adjacent pins and 6 wires under the module. The model for this is shown in Fig. 5.3.1.

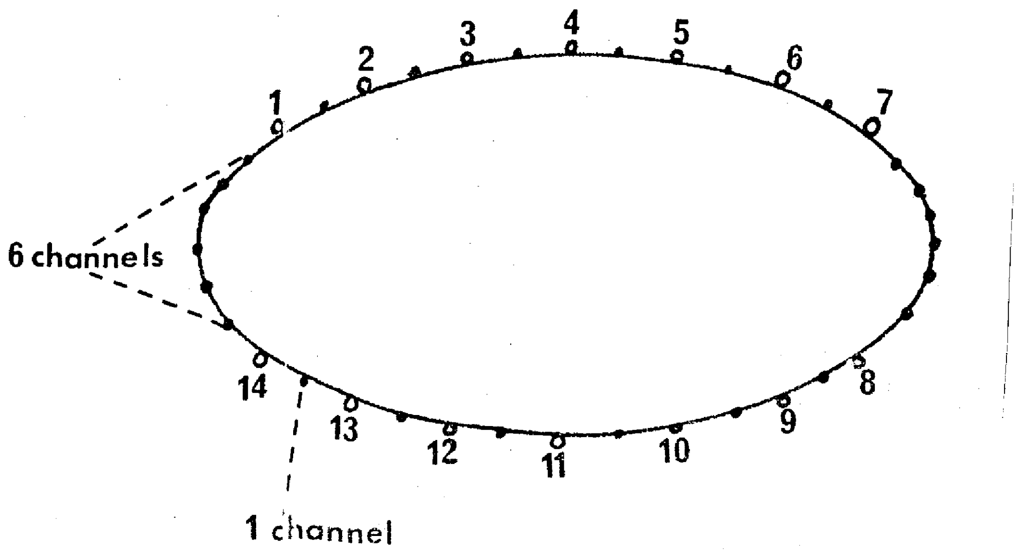


Fig. 5.3.1 Model for a 14 pin dual-in-line package with 1 channel between pins and 6 channels longitudinally.

In order to show how routing can be performed by using this model, a simple example is given in Fig. 5.3.2:

The circuit consists of two 6-pin packages A and B, allowing 1 wire between adjacent pins and 2 wires underneath (longitudinally). Assume that the following

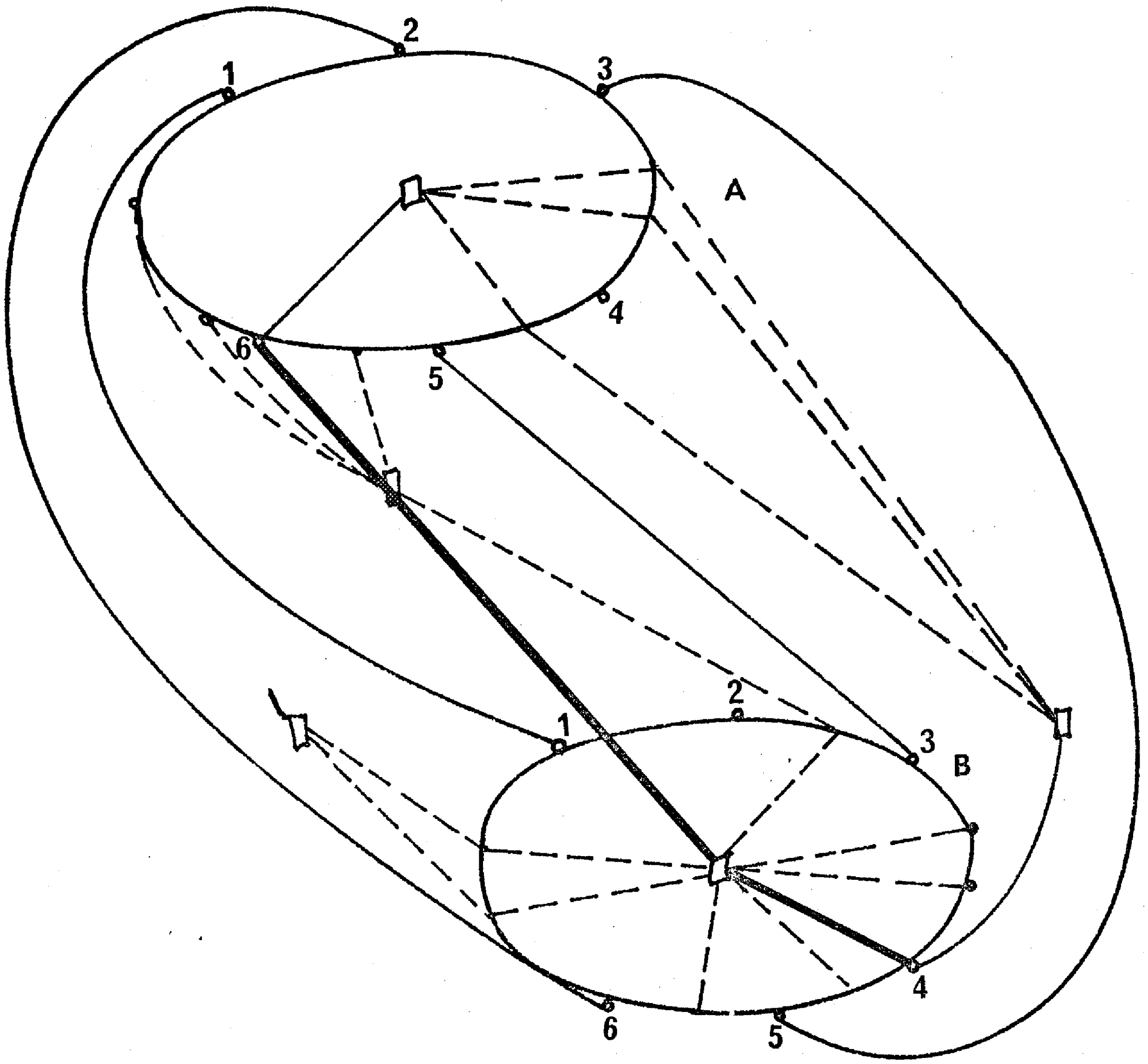


Fig. 5.3.2 Example of routing connection B4 - A6.

connections were routed through the use of an embedding algorithm: A1-B1, A2-B6, A3-B5, A5-B3. The connection A6-B4 is the next to be routed. The algorithm consists of the following steps:

1) Place inside each face of the graph a vertex.

2) Connect each 'center' vertex to all 'free' vertices in the corresponding face (by 'free' vertices is meant the vertices that represent the finite wiring capacities and that have not been used yet).

3) Connect the 'end' vertices (in this cases A6 and B4) to the 'center' vertices of the adjacent faces.

4) Consider the new graph so constructed (consisting of center, free and end vertices plus connecting edges) and apply a shortest path algorithm (weights can be assigned to the edges in order to optimize a parameter such as wirelength).

5) The vertices in this path are alternating center vertices and other. Remove the edges of the path and connect every second vertex by a new edge. This path forms the connection.

6) The edges of the new path divide certain faces into two new faces; delete the center vertex and incident edges for each old face and construct a new 'star' for the two new faces.

7) go to 3 to route the next connection.

5.4 Using Other Properties.

Sometimes it can be useful to group a maximal number of components together. This is for example the case in LSI layout. This grouping can be attempted after the topological embedding phase by considering the faces representing component boundaries and determining whether they are adjacent.

As an example of this, consider the circuit of Fig. 4.3.2. Assuming that gates are treated as individual components, the graph model for this circuit and its maximal planar subgraph are shown in Fig. 4.3.3 and 4.3.4 respectively.

By looking at the faces of the graph, it can be detected that components C3, A2, C1, B3 and C2 all appear on the same face and in that order. This allows us to join them into a single macro component, as illustrated in Fig. 5.4.1.

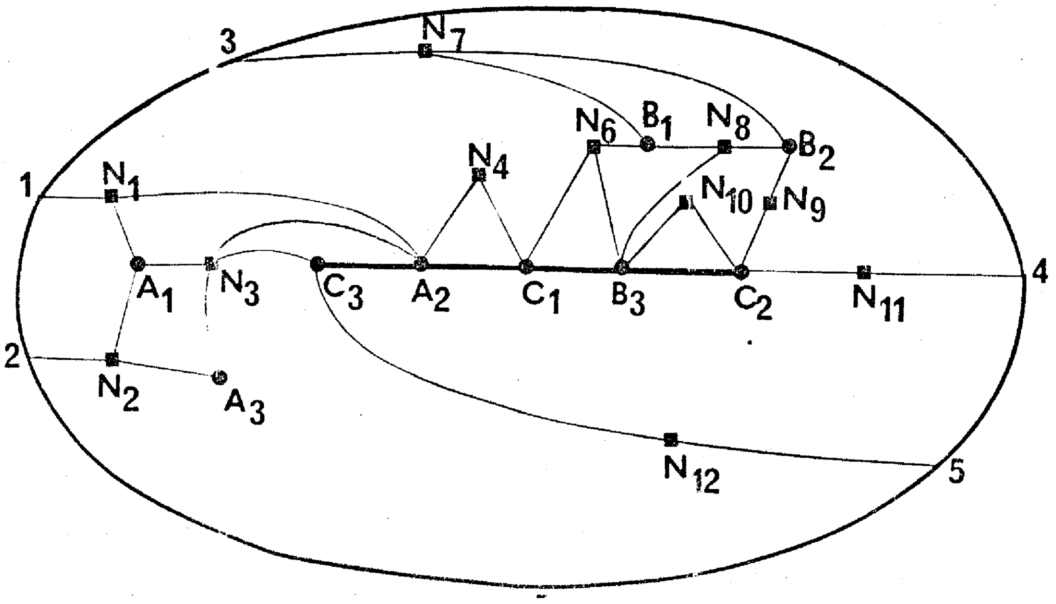


Fig. 5.4.1 Merging components C3, A2, C1, B3 and C2 into a single macro component.

Furthermore, components A1 and A3, as well as components B1 and B2 can be joined together, thereby reducing the circuit to three macrocomponents, without changing the embedding, as shown in Fig. 5.4.2.

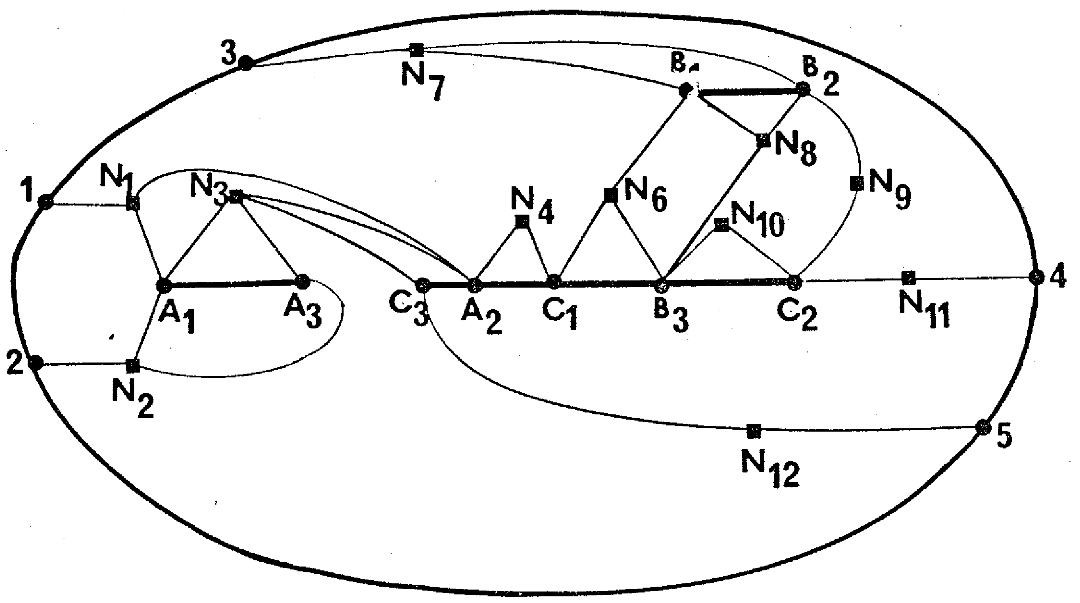


Fig. 5.4.2 Merging components A1,A3 and B1,B2 into macro components.

In section 3.5, it was shown that a K(1,n) subgraph was an adequate model for an n-terminal net if and only if no terminal of the net is a point of articuation of a component model and all connections are made to the region external to this model. This does not hold when one of the terminals in a net belongs to a set of physically equivalent terminals of a component. When a net, that is incident to a set of physically equivalent terminals, is not completely embedded, then one can attempt to connect some of the unconnected terminals of the net to the unconnected terminals of the physically equivalent set.

The models, derived in chapter 3, did not allow connections to be embedded inside a component's boundary. When this is allowed technologically, this property may be used to realize some of the unconnected nets.

5.5 Obtaining a Physical Embedding.

5.5.1. Introduction.

An important step in a circuit layout procedure is to obtain a physical embedding of the circuit graph. This embedding should take into account the geometrical properties of the circuit. Deriving a physical embedding is subject to the following constraints:

a) since the geometry of the components is predetermined, all vertices and edges, representing a component, have to be embedded with these geometrical constraints in mind.

b) all edges of the graph have to be embedded within a Manhattan geometry, i.e. only vertical and horizontal line segments are allowed.

c) a number of external connections has to be placed on the periphery of the drawing in prespecified locations.

An attempt to solve the problem of embedding a planar graph was made in (Ho71), although the constraints (a) and (b), related to circuit graphs, were not taken into account. Therefore, this attempt was not succesful for embedding circuit layout graphs. The system described in (Ho71) also requires a considerable amount of human interaction on an interactive graphics display. This is quite feasible for small circuits, but for large scale circuits, interaction should be minimized where possible.

5.5.2 An Algorithm for Obtaining a Preliminary Physical Embedding of Circuit Layout Graphs.

The circuit graph considered here has the following properties:

- every component is modelled by a cycle, representing its physical boundary.
- the pins (terminals) of a component are represented by vertices on this cycle.
- some components allow a finite number of interconnections to be routed between physically adjacent terminals. This property is also modelled by vertices on the cycle, with every vertex representing a routing capacity of exactly one interconnection. After the topological layout phase, these vertices are indistinguishable from those representing the component's terminals.

The result of the topological layout phase is a list of oriented faces of the graph.

The problem now is to transform this topological embedding into a physical one. In a physical embedding, the geometrical properties of the circuit are taken into account. The relative position of the terminals of a component is now a well-defined physical distance. The same is true for the external connections and for the physical size of the circuit.

The basic outline of the algorithm is as follows:

1) Find the inside face of the graph. [Start by labeling all faces adjacent to the peripheral cycle with a 1. Then, label with a 2 all faces not yet labeled that are adjacent to those with a label 1. Continue to do this until all faces are labeled. Select one of the faces with the highest label as the inside face.]

2) Embed the peripheral cycle as a rectangle with the external connections placed in the prescribed locations.

3) Break down the face into chains of one of the following types:

a) already embedded chains.

b) component periphery chains.

c) net (interconnection) chains.

4) Let the plane be divided into a number of squares, called "slots", large enough to contain the largest component. Place each of the components, for which there is a type b chain in the current face, into a slot and embed the cycle, representing this component.

5) Consider each of the net chains: if no part of the net has been embedded so far, find an interconnection path that satisfies the prescribed orientation of the face. If a part of the net has been embedded, find an interconnection between the start-vertex of the chain and all vertices and pseudo-vertices † of the net embedded so far. Select the shortest of the paths so obtained.

---

† (*) A pseudo-vertex is a point on the physical embedding of an edge, where two orthogonal line segments join.

6) If all faces have been embedded, stop; else, find the face, adjacent to the faces already embedded, that is the closest to the inside face. Go to 2.

While routing the net-chains, it is important that they be embedded in a well-specified order, such that the physical embedding corresponds to the topological embedding. The algorithm, used for routing these net-chains, is based on a line-searching algorithm by Hightower (Hi69). Its advantages are fast execution and minimal storage requirements, while its disadvantage is that it does not always find a path. The reason for the routing algorithm to fail is that the routing is performed on a finite resolution grid. By routing one connection, one might block the only available path for a connection to be routed later. A careful implementation reduces this blocking to a minimum.

Hightower's algorithm is well suited for this problem. In a normal routing problem, the algorithm requires storing and searching lists of already embedded vertical and horizontal line segments. These lists normally grow with the number of interconnections routed, making the algorithm less efficient while the routing proceeds. In this case, however, the drawing grows from the inside out. At any given stage, there is a peripheral cycle of the drawing, corresponding to the sum modulo 2 of all faces embedded at that time. All connections already routed, that are on the inside of the peripheral cycle, need not be searched since any new connection being routed can interact only with the current peripheral cycle. This property improves the speed of the routing algorithm, especially for large problems.

In a Manhattan geometry, no vertex can have a valency greater than 4. In the topological embedding phase, it is ensured that all vertices on component boundaries have a valency of at most 4. A vertex on a component boundary is incident to exactly 2 edges modelling the boundary and to at most 2 edges modelling the nets. The vertices modelling the nets, however, can have any number of incident edges. Fortunately, nets can be embedded as any spanning tree on the component vertices it connects. The method, described in step 5 of the algorithm usually results in adequate results. By slightly modifying the routing algorithm, one could try to find the shortest Manhattan path between a given vertex and the already embedded part of the net.

however, the drawing grows from the inside out. At any given stage, there is a peripheral cycle of the drawing, corresponding to the sum modulo 2 of all faces embedded at that time. All connections already routed, that are on the inside of the peripheral cycle, need not be searched since any new connection being routed can interact only with the current peripheral cycle. This property improves the speed of the routing algorithm, especially for large problems.

In a Manhattan geometry, no vertex can have a valency greater than 4. In the topological embedding phase, it is ensured that all vertices on component boundaries have a valency of at most 4. A vertex on a component boundary is incident to exactly 2 edges modelling the boundary and to at most 2 edges modelling the nets. The vertices modelling the nets, however, can have any number of incident edges. Fortunately, nets can be embedded as any spanning tree on the component vertices it connects. The method, described in step 5 of the algorithm usually results in adequate results. By slightly modifying the routing algorithm, one could try to find the shortest Manhattan path between a given vertex and the already embedded part of the net.

As an example of applying this algorithm, consider the maximal planar subgraph, shown in Fig. 5.2.1. The preliminary physical layout, obtained from the topological embedding is shown in Fig. 5.5.1.

It can be seen easily that this preliminary layout results in a very inefficient use of the area available. A simple technique for reducing the physical area of the circuit consist of squeezing together the preliminary layout, similar to the way prposed in (AG70). The result of this operation is shown in Fig. 5.5.2.

5.6 Summary.

In this chapter, we have suggested a number of algorithms for completing the topological layout by using some of the technology-dependent properties.

Furthermore, a algorithm was developed, for deriving a preliminary physical layout from the topological alyout. When a second layer of interconnections is needed, one could apply a classical routing algorithm (e.g. Hi69), to connect the remaining nets in this second layer.

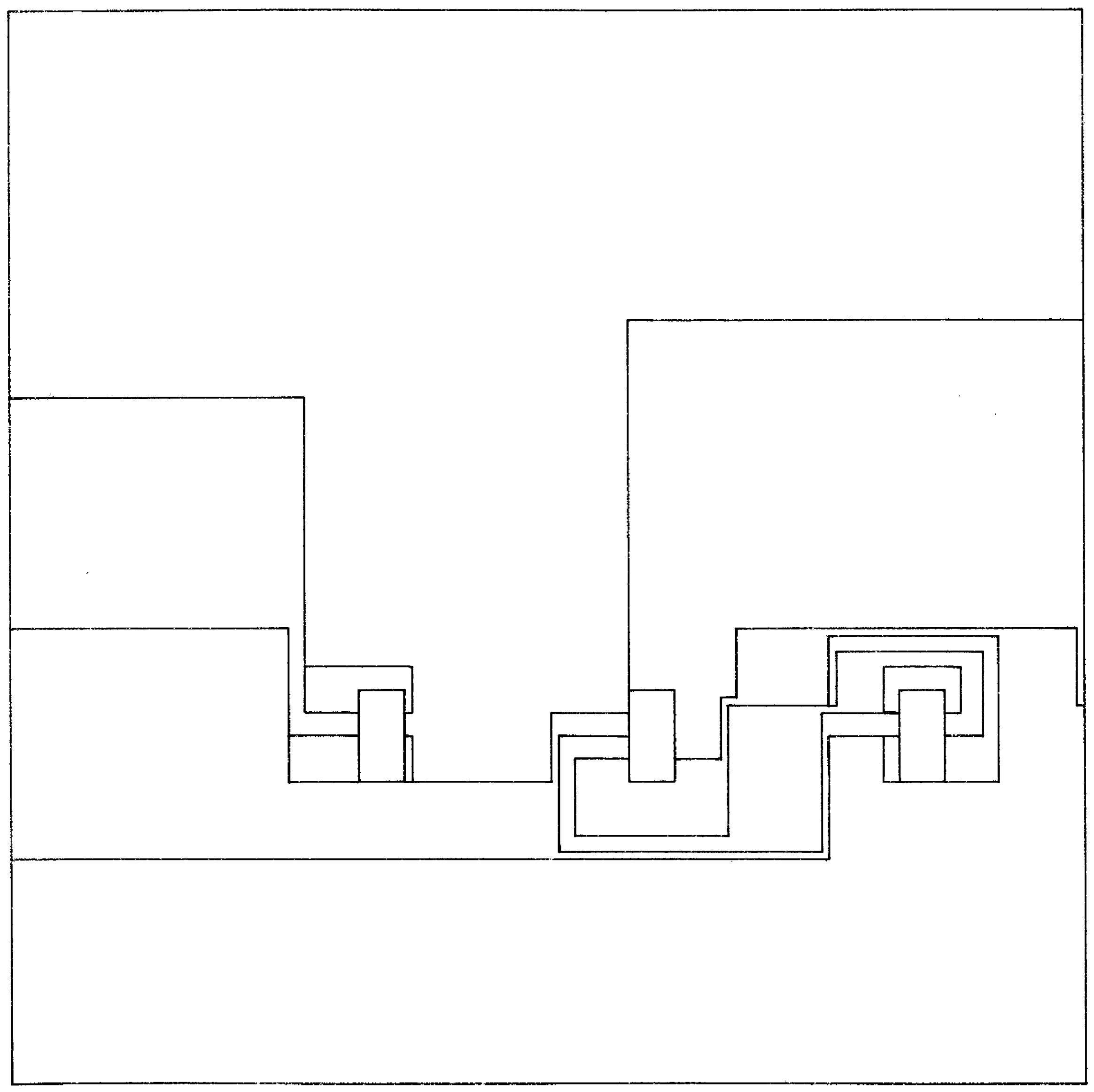


Fig. 5.5.1 Preliminary physical layout of the graph of Fig. 5.2.1.

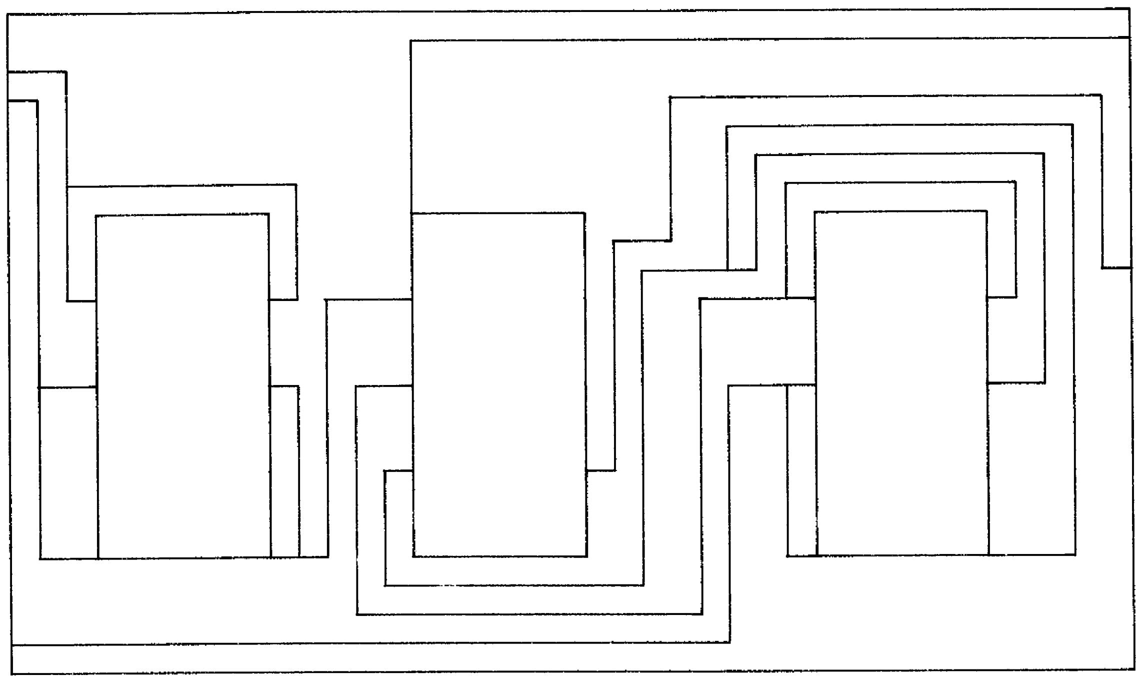


Fig. 5.5.2 Condensed physical layout.

6. Conclusions and Suggestions for Further Research.

6.1 Conclusions.

The main contribution in this dissertation is to provide a better mathematical model for the formulation and the solution of the circuit layout problem. The concepts of physical and logical equivalence of terminals and subcomponents were introduced in order to describe the mathematical properties of physical circuits.

Using these properties, greatly improved graph models for the circuit layout problem were developed in chapter 3. It is hoped that this mathematical formulation will lead to a better understanding of the problem.

A number of working systems for the layout of integrated circuits make use of topological layout methods (e.g. EM73, BG74, BR74, Kl73, Ko75, Su74). However, these systems are limited to 2 and 3-terminal devices and rely heavily on interactive graphics for obtaining a workable solution. The actual problems, solved by these systems, are usually small-scale bipolar integrated circuit layouts, with fewer than 100 components.

The models developed in this thesis allow multiterminal devices with a variety of physical properties to be used in a topological layout procedure. Furthermore, the algorithms proposed in chapters 4 and 5 require far less human interaction.

Because of these characteristics, the methods developed in this thesis are especially suited for the layout of large-scale integrated circuits.

6.2 Suggestions for Further Research.

6.2.1. The problem of finding a minimum set of edges to be removed from a non-planar graph to make the remaining graph planar has no known efficient solution. It would be useful to determine whether this problem is NP-complete or whether this problem can be solved in polynomial time.

6.2.2 Assuming that the maximum planar subgraph problem cannot be solved in polynomial time, it would be desirable to develop efficient polynomial-time algorithms for this problem that give an approximate solution and to provide a lower bound of how 'optimal' this solution is, both in the general case and in the case of circuit layout graphs.

6.2.3. Find an algorithm to partition a circuit graph into 2 or more planar subgraphs with edge sets of almost equal cardinality. Note that the "pseudo-edges" that represent the modules have to be repeated in each subgraph. If possible, the separation should be done at the vertices that represent the pins.

6.2.4 In chapter 4, it was pointed out that embedding a circuit layout graph is not the same as simply partitioning the graph into a minimum number of planar subgraphs. Furthermore, the embedding has to satisfy a number of constraints.

We could define a circuit layout graph to be _k-planar_ if it can be embedded without crossings on k layers but not on k-1 layers. The approach taken in chapter 4, where a maximal planar subgraph is found first and embedded on the first layer, might not necessarily lead to an embedding in a minimum number of layers. It would be useful if some characterization of k-planar circuit layout graphs could be found.

Some interesting work on embedding graphs in multiple layers with certain boundary conditions (not related to the circuit layout problem) was reported by Bernhardt and Kainen (BK00).

6.2.5 In chapter 3, it was mentioned that there exists an hierarchical relationship between the physical properties of circuits. This relation induces a lattice structure on the set of terminals. Further research into this may lead to a better understanding of the circuit layout problem.

6.2.6 When designing large scale integrated circuits, the number of single components becomes very large ($O(10^3)$). This would result both in excessive time and space requirements. Usually, the circuit can be functionally partitioned into 'macro-cells' such as NAND gates, flip flops, etc.. Some existing LSI layout procedures, make use of a library of predesigned functional blocks. When an automatic layout procedure is available, it might become feasible to redesign a cell topologically, in function of an optimal layout. The graph model for such a cell would contain as much freedom as possible under technological constraints. The layout procedure would then provide the order in which terminals have to appear on a component's boundary. Using this information, a topological layout of the individual cell can then be performed.

6.2.7 The problem of minimizing an IC chip's area is very important, especially for designs that are to be produced in large volume. Assuming that the topological layout has been done, one could perform a physical layout of every cell in function of an optimal use of the chip's area. This would probably result in very different physical shapes for cells of the same type. The approach proposed here and in 6.2.6 actually emulates, to some extent, the steps a human designer would go through when trying to design an integrated circuit with a minimal area.

Bibliography and References.

[Ab72] Abel,L.C. "On the Automated Layout of Multi-Layer Planar Wiring and a Related Graph Coloring Problem.", University of Illinois, Dept. of Computer Science, Ph.D. Thesis, Coord. Science Lab. (CSL) Report R-546, Jan. 1972.

[Ak67] Akers,S.B. "A Modification of LEE'S Path Connection Algorithm.", IEEE Trans. on Electronic Computers, Vol. EC-16,Feb. 1967, pp. 97-98.

[Ak72] Akers,S.B. "Routing.", in: Breuer,M.A. : Design Automation of Digital Systems, Prentice Hall, 1972 (Ch 6).

[AG70] Akers,S.B., Geyer,J.M. and Roberts,D.L. "IC Mask Layout with a Single Conductor Layer.", Proc. 7-th Design Automation Workshop, San Francisco, June 1970, pp. 7-16.

[AH69] Akers,S.B. and Hadlock,F.O. "Graph Theory Models of Electrical Networks and their Minimum Cross-over Layouts.", Conf. on Graph Theory and Computing, Kingston, Jamaica, Jan. 1969.

[AT59] Auslander,L. and Trent,H.M. "On the Topology of Printed Circuits.", IRE Trans. on Circuit Theory, Vol. CT-6 No. 4, pp. 390-391, Dec. 1959.

[Ba64] Bader,W. "Das Topologische Problem der Gedruckten Schaltung und seine Losung (German - the Topological Problem of Printed Circuits and its Solution).", Archiv fuer Elektrotechnik, Vol. 49, pp. 2-12, 1964.

[BN73] Basden,A. and Nichols,K.G. "New Topological Method for Laying out Printed Circuits.", Proc. IEE, Vol. 120, No. 3, pp. 325-328, March 1973.

[BG74] Bastin,D., Girard,E., Rault,J.C., Rosenstiehl,P. and Souesme,P. "Le Dessin Automatique des Masques de Circuits Integres: un nouvel Algorithme d'Implantation.", Revue Technique Thomson-CSF, Vol. 6, No. 1, pp. 229-246, March 1974.

[BR74] Baubock,E. and Renelt,G. "Rechnerunterstuetzter Entwurf von Layouts Integrierter Schaltungen.", Valvo Berichte, Vol. 18, No. 1/2, pp. 189-202, 1974.

[Be67] Bedrosian,S.D. "Topological Guides to Integrated RC Circuit Design.", Record 10th Midwest Symp. on Circuit Theory, Purdue University, 1967, Session VI.1, 12 pp.

[BC71] Behzad,M. and Chartrand,G. "Introduction to the Theory of Graphs.", Boston: Allyn and Bacon, 1971.

[Be72] Berge,C. "Graphs and Hypergraphs.", New York: North American Elsevier, 1972, 555 pp..

[BK00] Bernhardt,F. and Kainen,P.C. "On the Book-thickness of a graph.", to appear.

[BU68] Bittner,B. and Ulrich,J. "A Method for Improving the Location of Connecting Paths for Circuit Card Wiring.", IEEE Computer Society Repository, R68-128, 1968, 16 pp..

[Br74] Brehaut,W.B. "On Planar Graphs and the Plainer Nonplanar Graphs.", University of Waterloo, Ph. D. Thesis, 1974.

[Br75] Brehaut,W.M. "Efficient Planar Mesh Finding.", submitted for publication, 1975.

[Br72] Breuer,M.A. (ed.) "Design Automation of Digital Systems.", Englewood Cliffs, N.J.: Prentice Hall, 1972.

[Ch70] Chein,M. "Sur les Problemes de Decomposition d'un Graphe, lies a l'Implantation.", These de Doctorat es Sciences, Universite de Grenoble, june 1970.

[Da69] Danielson,G.H. "Applications of Linear Graph Theory to Automated Networks Layout.", Ph.D. Thesis, Syracuse University, 1969, 125 pp; Diss. Abstr. Vol. 30, No. 12, p. 5494B ; University Microfilms No. 70-10334. also: Report R69ELS73, General Electric Co, Electronics Lab., Syracuse.

[DM64] Demoucron,G., Malgrange,Y. and Pertuiset,R. "Graphes Planaires: Reconnaissance et Construction de Representations Planaires Topologiques.", Revue Francaise de Recherche Operationnelle, Vol. 8, pp 34-47, April 1964.

[De71] Dennis,M.G. "The Layout Problem for Graphs.", Harvard University, Electronic Systems Division, Report ESD-TR-71-344 (ARPA Order # 952), August 1971.

[EM69a] Engl,W.L. and Mlynski,D.A. "A Graph Theoretical Layout Procedure for Integrated Circuits.", Int. Symp. on Circuit Theory 1969 , Digest pp. 37-39.

[EM69b] Engl,W.L. and Mlynski,D.A. "Topological Synthesis Procedure for Circuit Integration.", ISSCC 69 - Digest of Technical Papers, pp. 138-139.

[EM70] Engl,W.L. and Mlynski,D.A. "An Algorithm for Embedding Graphs in the Plane with Certain Constraints.", IEEE Trans. on Circuit Theory, May 1970, pp. 250-252.

[EM72a] Engl,W.L. and Mlynski,D.A. "Mengentheorie Verallgemeinter Graphen.", Archiv fuer Elektrotechnik, Vol. 54, No. 5, 1972, pp. 278-284.

[EM72b] Engl,W.L. and Mlynski,D.A. "Die Schaltungsintegration als Graphentheoretisches Syntheseproblem.", Archiv fuer Elektrotechnik, Vol. 54, pp. 315-324, 1972.

[EM72c] Engl,W.L. and Mlynski,D.A. "Die Losung des Problems der Topologischen Schaltungsintegration.", Archiv fuer Elektrotechnik, Vol. 54, pp. 325-336, 1972.

[EM72d] Engl,W.L. and Mlynski,D.A. "Topological Theory of Integrated Circuits.", Proc. NATO Symposium on Network Theory, London, 1972, 10pp.

[EM73] Engl,W.L., Mlynski,D.A. and Pernards,P. "Computer Aided Topological Design for Integrated Circuits.", IEEE Trans. on Circuit Theory, Vol. CT-20, No. 6, pp. 717-725, Nov. 1973.

[EM75] Engl,W.L., Mlynski,D.A. and Pernards,P. "Theory of Multiplace Graphs.", IEEE Trans. on Circuit and Systems, Vol. CAS-22, No. 1, pp. 2-8, Jan. 1975.

[FW64] Fisher,G. and Wing,O. "An Algorithm for Testing Planar Graphs from the Incidence Matrix.", Proc. 7th Midwest Symp. Circuit Theory, Ann Arbor, May 1964, pp. 67-75.

[FW66] Fisher,G. and Wing,O. "Computer Recognition and Extraction of Planar Graphs from the Incidence Matrix.", IEEE Trans. Circuit Theory, vol. CT-13, pp. 154-163, June 1966.

[FC67a] Fisk,C.J., Caskey,D.L. and West,L.E. "Topographic Simulation as an Aid to Printed Circuit Board Design.", Proc. 4-th Design Automation Workshop, Los Angeles, June 1967, pp. 17.1-17.23.

[FC67b] Fisk,C.J., Caskey,D.L. and West,L.E. "ACCEL : Automated Circuit Card Etching Layout.", Proc. IEEE, Vol. 55, No. 11, pp. 1971-1982, Nov. 1967.

[Fl72] Fletcher,A.J. "EUREKA: a system for laying out Circuits using a single Layer of Interconnections.", Proc. Int. Conference on Computer-aided Design, University of Southampton, April 1972, p 25-30.

[Fu67] Fu,Y. "Application of Linear Graph Theory to Printed Circuits.", Proc. Asilomar Conf. on Systems and Circuits, Nov..1967, p 721-728 (Also CFSTI AD 676-711).

[GJ74] Garey,M.R., Johnson,D.S. and Stockmeyer,L. "Some simplified NP-complete Problems.", Proc. 6th Annual ACM Symposium on the Theory of Computing, Seattle, April 1974, pp. 47-63.

[GO63] Goldstein,A.J. "An effiecient And constructive Algorithm for testing whether a Graph can be imbedded in a plane.", Princeton Univ., 1963, 2 pp.

[GS73] Goldstein,A.J. and Schweikert,D.G. "A Proper Model for Testing the Planarity of Electrical Circuits.", BSTJ, Vol. 52, No. 1, pp. 135-142, Jan. 1973.

[HK70a] Hanan,M. and Kurtzberg,J.M. "Force-Vector Placement Techniques.", IBM Research Report RC 2843, April 1970.

[HK70b] Hanan,M. and Kurtzberg,J.M. "Placement Techniques.", IBM Research Report RC 2846, April 1970.

[HK70c] Hanan,M. and Kurtzberg,J.M. "A Review of the Placement and Quadratic Assignment Problems.", IBM Research Report RC 3046, April 1970.

[HK72a] Hanan,M and Kurtzberg,J.M. "Placement Techniques.", M.A. Breuer, Design Automation of Digital Systems, (Ch 5), Prentice Hall, 1972.

[HK72b] Hanan,M. and Kurtzberg,J. "A Review of the Placement and Quadratic Assignment Problems.", SIAM Review, Vol. 14, No. 2, pp. 324-342, April 1972.

[Ha60] Halmos,P.R. "Naive Set Theory.", Princeton, New Jersey: Van Nostrand Cy., 1960.

[Ha62] Harary,F. "A Complementary Problem on Nonplanar Graphs.", Math. Mag., Vol. 35, pp. 301-303, 1962.

[Ha69] Harary,F. "Graph Theory.", Reading, Mass.: Addison Wesley Publishing Cy., 1969.

[HN71] Hashimoto,A. and Noshita,K. "A Property of a N-Graph.", IEEE Trans. on Computers, Vol. C-20, No. 1, pp. 95-97, Jan. 1971.

[HS71] Hashimoto,A. and Stevens,J. "Wire Routing by Optimizing Channel Assignment within Large Apertures. Proc. 8th Design Automation Workshop, Atlantic City, June 1971, pp. 155-169.

[Hi69] Hightower,D. "A Solution to Line Routing Problems in the Continuous Plane.", Proc. 6th Design Automation Workshop, 1969, pp 1-24.

[HT73] Hopcroft,J. and Tarjan,R. "Efficient Planarity Testing.", Cornell University, Dept. of Computer Science, Report TR 73-165, April 1973.

[HT74] Hopcroft,J. and Tarjan,R.E. "Efficient Planarity Testing.", J. ACM, vol. 21, no. 4, pp. 549-568, Oct. 1974.

[Ho71] Hope,A.K. "A Planar Graph Drawing Program.", Software, Practice and Experience, Vol. 1, pp. 83-91, 1971.

[Ka72] Karp,R.M. "Reducibility among Combinatorial Problems.", University of California, Berkeley, Dept. of Computer Science, Technical Report No. 3, April 1972.

[Kl73] Klamet,H. "Computer-Aided Design of the Layout of Integrated Circuits (CADLIC).", Proc. International Computing Symposium, Davos, Switzerland, 1973, pp. 451-458.

[Ko61] Kodres,U.R. "Formulation and Solution of Circuit Card Design Problems through Use of Graph Methods.", Proc. IECP Symp., Boulder, Colorado; Advances in Electronic Circuit Packaging, G.A. Walker, Ed., Vol. 2, pp. 121-142, 1961.

[Ko69] Kodres,U.R. "Logic Circuit Layout.", Proc. Joint Conf. on Mathematical and Computer Aids to Design, Anaheim, California, Oct. 1969, pp. 165-191.

[Ko72] Kodres,U.R. "Partitioning and Card Selection.", in: M.A. Breuer, "Design Automation of Digital Systems.", Englewood Cliffs: Prentice Hall, 1972, pp. 173-212.

[KO75] Koppe,R. "Das Abbildungsproblem beim rechnerunterstuetzter Entwurf von Layouts integrierter Schaltungen.", Angewandte Informatik, vol. 17, no. 6, pp. 223-232, June 1975.

[Koz72] Kozawa,T. et al. "Block and Track Method for Automated Layout Generation of MOS-LSI Arrays.", Digest ISSCC, Philadelphia, February 1972, p 62-63.

[La69] Lass,S.E. "Automated Printed Circuit Routing with a Stepping Aperture.", Comm. ACM, Vol. 12, No. 5, pp. 262-266, May 1969.

[Le61] Lee,C.Y. "An Algorithm for Path Connections and its Applications.", IRE Trans. on Electronic Computers, Vol. EC-10, pp. 346-365, 1961.

[LE66] Lempel,A., Even,S. and Cederbaum,I. "An Algorithm for Planarity Testing of Graphs.", Proc. Int. Symp. on the Theory of Graphs (Ed. P. Rosenstiehl), Paris: Dunod, 1966, pp. 215-232.

[LM64] Lerda,F. and Majorani,E. "An Algorithm for Connecting N Points with a Minimum Number of Crossings.", Calcolo, Vol. 1, pp. 257-265, 1964.

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

[LL71] Levi,G. and Luccio,F. "A Weighted Graph Embedding Technique and its Application to Automatic Circuit Layout.", Calcolo, Vol. 8, No. 1-2, pp. 49-60, 1971.

[LL73] Levi,G. and Luccio,F. "A Technique for Graph Embedding with Constraints on Node and Arc Correspondences.", Information Sciences, Vol. 5, pp. 1-24, 1973.

[Ma64] Majorani,E. "Simplification of Lee's Algorithm for Special Problems.", Calcolo, Vol. I, pp. 247-256, 1964.

[MP73] Mlynski,D.A. and Pernards,P. "Planarization of Circuit Graphs.", Proc. Hawai Conf. on System Sciences, Jan. 1973, pp. 208-211.

[Na72] Nakahara,H. "Computer Aided Interconnection Routing: General Survey of the State of the Art.", Networks, Vol. 2, No. 2, pp. 167-183, 1972.

[PY70] Parkin,R.E. and Yaged,B. "On the Layout of Thin-Film RC Circuits for few Cross-overs.", Proc. Asilomar Conf. on Circuits and Systems Sciences, 1970, pp. 126-130.

[Pl70] Plisch,D.C. "New Results Concerning Separation Theory of Graphs.", Ph.D. Thesis, University of Wisconsin, 1970, 220 pp.; Diss. Abstr., Vol. 31, No. 11, p. 6616B; University Microfilms 71-2234.

[PB70] Plisch,D.C. and Brown,D.P. "Some New Results Concerning Separation Theory of Graphs.", Proc. 8th Allerton Conf. on Circuit and System Theory, 1970, pp. 505-514.

[Ra72] Raymond,T.C. "Determining Printed Circuit Wiring.", IBM Technical Disclosure Bulletin, Vol. 14, No. 11, pp. 3523-3525, April 1972.

[Re70] Read,R.C. "Graph Theory Algorithms." in: B. Harris, "Graph Theory and its Applications.", New York: Academic Press, 1970, pp. 51-78.

[Ro70] Rose,N.A. "Computer-Aided Design of Printed Wiring Boards.", Ph.D. Thesis, University of Edinburgh, April 1970.

[RO71] Rose,N.A. and Oldfield,J.V. "Printed-Wiring Board Layout by Computer.", Electronics and Power, Vol. 17, pp. 367-369, Oct. 1971.

[Ru72] Rubin,F. "Printed Wire Routing for Multilayer Circuit Boards.", Ph.D. Thesis, Syracuse University, 1972, 308 pp; Diss. Abstr., Vol. 33, No. 9, p. 4236B; University Microfilms 73-7767.

[Ru75] Rubin,F. "An Improved Algorithm for Testing the Planarity of a Graph.", IEEE Trans. on Computers, Vol. C-24, No. 2, pp. 113-121, Feb. 1975.

[Sa68] Saito,M. "A Condition for the Planarity of Electronic Circuits.", Electronics and Communications in Japan, Vol. 51, No. 8, pp. 166-167, Aug. 1968.

[SK72] Schweikert,D.G. and Kernighan,B.W. "A Proper Model for the Partitioning of Electrical Circuits.", Proc. 9th Design Automation Workshop, Dallas, June 1972, pp. 57-62.

[Sh69] Shirey, R.W., "Implementation and Analysis of Efficient Planarity Testing Algorithms.", Ph.D. Thesis, University of Wisconsin, 1969.

[Si66a] Sinden,F.W. "Topology of Thin Film RC Circuits.", BSTJ, Vol. 45, Nov. 1966, pp. 1639-1662.

[Si66b] Sinden,F.W. "Topology of Thin-Film RC Circuits.", in: Rosenstiehl,P. "Theory of Graphs", Paris: Dunod, 1966.

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

[St72] Stevens, J.E. "Fast Heuristic Techniques for Placing and Wiring Printed Circuit Boards.", Ph.D. Thesis, University of Illinois, Urbana-Champaign, 1972, 126 pp.; Diss. Abstr., Vol. 34, No. 2, p. 629B; University Microfilms, No. 73-17438.

[Su74] Sugiyama,N et al. "Integrated Circuit Layout Design System.", Computer Aided Design, Vol. 6, No. 2, pp. 66-72, 1974.

[SN70] Sugiyama,N., Nemoto,S., Kani,K., Ohtsuki,T. and Watanabe,H. "An Integrated Circuit Layout Design Program based on a Graph Theoretical Approach.", Digest ISSCC, Philadelphia, Feb. 1970, pp. 86-87.

[Ta71] Tarjan,R.E. "An Efficient Planarity Algorithm.", Ph.D. Thesis, Stanford University, Nov. 1971, 160 pp; Report CS-244-71; Diss. Abstr., Vol. 32, No. 12, p. 6974B; University Microfilms, No. 72-16807.

[Ul68] Ulrich,J.W. "A Computational Theory of Planar Imbedding.", Ph.D. Thesis, University of Texas, Austin, 1968. Diss. Abstr., Vol. 19, pp. 3840B; University Microfilms 69-6231.

[Ul69] Ulrich,J.W. "The Effect of Connector Orientation on the Physical Design of Electrical Networks.", Proc. Joint Conf. on Mathemathical and Computer Aids to Design, Anaheim, Oct. 1969, p. 412.

[Ul70] Ulrich,J.W. "A Characterization of Planar Oriented Graphs.", SIAM Journal Appl. Math, Vol. 18, No. 2, pp. 364-371, March 1970.

[US74] Uyehara,T., Shiraishi,H., Takahashi,O. and Kojima,T. "Embedding a Graph in a Plane with Local Constraints.", Proc. IEEE Int. Symp. on Circuits and Systems, San Francisco, April 1974, pp. 181-185.

[Va74a] vanCleemput.W.M. "Automated Design of Digital Systems - A Bibliography.", University of Waterloo, Dept. of Applied Analysis and Computer Science, Research Report 74-08, May 1974.

[Va74b] vanCleemput,W.M. "Topological Methods for the Circuit Layout Problem.", Proc. 8th Annual Princeton Conf. on Information Sciences and Systems, March 1974.

[Va74c] vanCleemput W.M. "The Use of Graph-Theoretical Methods for Integrated Circuit Design.", Int. Conf. on Computers in Engineering and Building Design, Imperial College, London, September 1974.

[VL74] vanCleemput W.M. and J.G. Linders "An Improved Graph Theoretical Model for the Circuit Layout Problem.", Proc. 11th Design Automation Workshop, Denver, June 1974.

[VO73] Vanlier,M.C. and Otten,R.H. "On the Mathematical Formulation of the Wiring Problem.", Int. J. of Circuit Theory and Applications, Vol. 1, pp. 137-147, 1973.

[We68] Weinberg,L. "Microelectronics and Printed Circuits: Problems and their Solutions.", Proc. 5-th Design Automation Workshop, Washington, July 1968, pp. 3.1-3.30.

[We62] Weissman,J. "Boolean Algebra, Map Coloring and Interconnections.", Amer. Math. Monthly, Vol. 69, pp. 608-613, 1962.

[YN69] Yoshida,K. and Nakagawa,T. "Topological Layout Design of Monolythic IC in CAD.", Digest ISSCC, Philadelphia, Feb. 1969, pp. 136-137.

[YO69] Yoshida,K. and Ohta "Topological Layout of a Monolythic IC.", Electronics and Communications in Japan, Vol. 52, No. 12, pp. 173-179, 1969.