設 計 自 動 化 52－8
（1990．5．22）

アナログ回路基板設計システムにおける極大平面グラフ抽出算法の性能評価

荒木知平　岩元圭一郎　渡辺敏正　翁長健治
広島大学　工学部
〒724　東広島市西条町大字下見

本論文では，アナログ回路等の一層プリント基板配線設計におけるジャンパー線の極少化問題に対する，制約条件を考慮した全域極大平面部分グラフを求める計算時間が$O(|V||E|)$の近似アルゴリズムを提案する．この近似アルゴリズムの特徴は，全域極大平面部分グラフを求めるに際して，ＰＱＲ－木を用いていることである．そして，極大平面部分グラフを求める二つの近似アルゴリズムの性能評価を行なった．

Finding a Minimal Set of Jumpers in the Design of Printed Wiring Boards for Analog Circuits

Tomohira Araki, Keiichiro Iwamoto, Toshimasa Watanabe, and Kenji Onaga
Faculty of Engineering, Hiroshima University,
Saijo-cho, Higashi-Hiroshima, 724 Japan
Phone: 0824-22-7111 Ext.3285(Watanabe) Fax: 0824-22-7195

The paper proposes an $O(|V||E|)$ algorithm for finding a minimal set of jumpers in the design of printed wiring boards of analog circuits. It finds, under certain physical conditions, a maximal spanning planar subgraph of a graph constructed from given net lists. The algorithm is based on the planarity testing using PQR-trees. Comparison of solutions given by two planarization algorithm are given.

1. Introduction

The paper proposes an $O(|V||E|)$ algorithm PLAN-PWB for finding a minimal set of jumpers in the design of printed wiring PLAN-PWB boards of analog circuits. Suppose we are given a set of net lists of an analog circuit under consideration, a placement of modules (such as ICs, registers and condensers) with their terminals (the spots where the pins are to be attached) located on one side of a board. What we are going to do is to find a maximal planar routing among those terminals on that board, or equivalently to find a minimal set of jumpers (lines not assigned to be attached to the board), under the following conditions (1)-(4):

(1) There are two kinds of modules; *up-sided* ones (each having a specified side which has to be faced to the board in actual placing) and *free* ones (otherwise).

(2) Modules have to be placed on one side of the board, where up-sided modules should be placed as specified.

(3) Routing through *module areas* (an area on the board to be occupied by one module) is prohibited. (We add this condition to simplify the discussion. There exist some modules which allow routing through their module areas: such one can be handled by means of post-processings.)

(4) Non-jumpers (lines assigned on the board) do not cross each other, while crossing of jumpers and non-jumpers are permitted as long as they do not violate physical requirements.

The problem is a kind of finding a maximal spanning planar subgraph of a given graph G=(V,E). Incorporation of the conditions (1)-(3), which come from practical requirements, show the difference between the previous maximal planarization problems[7,8,9,15] and this one. We consider the maximality, since it is known that the "maximum" planarization problem with the conditions (1)-(3) incorporated is NP-complete[14]. [13] proposed an $O(|V||E|)$ algorithm for finding a maximal spanning planar subgraph of a given graph, where none of the condition(1)-(3) is considered. They used the PQ-tree[9], while we use the PQR-tree[14], a variation of the PQ-tree, to handle those conditions. Our algorithm is obtained by adapting the algorithm of [13] for our subject. Although the adaptation is not so difficult, the result seems new and is very useful in the design of printed wiring boards for analog circuits, as is described briefly in the next section. The research of PLAN-PWB is motivated during the development of a printed wiring board design system PRIDE proposed in [17].

The main result of the paper is that we can determine in $O(|V|+|E|)$ time a "minimum" set of edges (jumpers) to be deleted whenever the reduction of the PQR-tree becomes impossible (that is, the current intermediate subgraph is nonplanar). This procedure will be repeated at most $O(|V|)$ times and, therefore, a "minimal" set of jumpers can be obtained in $O(|V||E|)$ time by the union of such "minimum" sets. [14] proposed an $O(|V|)$ algorithm for detecting the planarity of a graph constructed from a given net list under the conditions(1)-(3). A minimal set of jumpers can be given by repeating adding an edge followed by this planarity testing. We call this algorithm REPEAT-PLAN, and its time complexity is $O(|V||E|)$. Experimental evaluation through practical data (audio circuits) shows that PLAN_PWB proposed in the paper produces smaller solutions than REPEAT-PLAN.

2. Preliminaries

2.1. Graph theoretical Formulation

The problem is formulated graph-theoretically, according to [14]. Given a circuit represented by a set of net lists, we construct a graph G(C)=(V,E) as follows, where $E=E_0 \cup E_1 \cup E_2$:

1° Place one vertex for each terminal and one simple edge between each pair of terminals requiring their connection.

2° Represent each of two-terminal modules and free modules as a wheel with the terminals connected as a cycle having a new vertex inside (Fig. 2.1(a)).

3° Represent each of up-sided modules as a directed cycle considering of the terminals, where they appear the same as can be seen clockwise from above that module (Fig. 2.1(b)).

4° Let V be the set of vertices introduced in these steps. Let E_0, E_1 and E_2 devote the set of edges introduced in 1°, 2° and 3°, respectively, where every edge of E_2 is directed.

Wheels prevent routing through module areas, and directed cycles are used to force up-sided modules to placed as specified. G(C) is devoted by G for simplicity. The subject of the paper is stated graph-theoretically as follows:

"Find a maximal spanning planar subgraph G'=(V,E') of G satisfying (a) $E_1 \cup E_2 \subseteq E'$, and(b) there exists a plane embedding (a graph drawn on a plane without edge crossing) of G' such that every directed cycle is drawn clockwise without edges existing inside."

Maintaining clockwise directedness is done by incorporating R-nodes into PQ-trees (introduction of PQR-trees in [14]), and avoiding edges inside directed cycles is handled during the reduction process of PQR-trees.

It should be noted that a star-shaped connection in a circuit (see Fig. 2.2(a) showing a T-shaped connection) is represented as a complete connection requirement (Fig. 2.2(b)) among those terminals. This may increase the number of jumpers. After some jumpers are found, the designer can determine real jumpers among them, by specifying a spanning subtree of that complete connection.

2.2. PQR-trees

A PQR-tree, introduced in [14], is a directed ordered rooted tree consisting of four kinds of nodes, P-nodes, Q-nodes, R-nodes and leaves. Fig. 2.3 shows an example of a PQR-tree, where a circle, a rectangle without on arrow and one with an arrow denote a P-node, a Q-node and an R-node, respectively. (We use this notation throughout the paper.) All nodes except R-nodes are elements of well-known PQ-trees [9].

Let F(T) denote a sequence defined by concatenating leaves of a PQR-tree T from left to right. In Fig. 2.3, F(T)=abc. F(T) is called a *frontier* of T and represents a permutation. Two PQR-trees T and T' are *equivalent* (denoted by T≡T') if and only if T' is obtained from T by repeating the following two transformations (i) Changing the order of children of a P-node arbitrarily, and (ii) Reversing the order of children of a Q-node. We note that the order of children of any R-node cannot be changed. This is because an R-node is introduced to represent a directed cycle handling an up-sided modules. Let con(T)={F(T') | T'≡T}. A set S consisting of only leaves of T is called a *leaf set* of T. Given a leaf set S of T, a *reduction* of T for S is a procedure to construct a PQR-tree T' such that

con(T')={π|π ∈ con(T) and all elements of S appear in π consecutively}.

As in the case with PQ-trees, existing an executable reducibility means that adding some edges incident upon a certain vertex to the current subgraph preserve planarity. If no reduction is possible then the current subgraph is nonplanar, and we search the present PQR-tree for a "minimum" set of edges whose deletion recover planarity. Our searching method comes from the one using PQ-trees proposed in [13], with some adaptations incorporated in order to handle R-nodes. A reduction is done by one of template matchings as shown in Fig. 2.4 (1)-(10). They are taken from [14].

3. Maximal Planarization with PQR-trees

We describe an $O(|V||E|)$ algorithm PLAN-PWB for finding such a maximal spanning planar subgraph H of G as described in Section 3, by using PQR-trees. In this section we assume that G in a 2-connected graph (a graph without cutvertices) with V={1, ···, n}.

3.1. An algorithm PLAN-PWB.

It is noted that each step except Step 4 comes from those corresponding one of the algorithm in [13] by replacing the term "PQ-tree" with "PQR-tree" and by modifying appropriately. See Figures 3.1 and 3.2 for an example. The detail of Step 4 will be given later.

<PLAN-PWB>

1. Apply the st-numbering algorithm[10] to G, Where, for simplicity, we consider the vertex i∈ V, is the i-th st-numbered one. construct a PQR-tree T consisting of only the vertex 1. N←1.
2. (1) Construct a PQR-tree T_N for the vertex N as shown in Fig. 3.3 (a)-(d), where directed edges are handled carefully t avoid edges inside directed cycles. {Leaves of T_N correspond to vertice adjacent to N in G. In the construction of PQR-trees, if d edges are incident upon a vertex v in G then we provide d copies of v. These copies appear as leaves of PQR-trees T_u, where u is adjacent to v, and we try to coalesce them in the following reductions.}
   (2) Delete all copies of the vertex N appearing as leaves of T, and add T_N into T by making the root of T_N as a child of the node to which those deleted copies were adjacent. Let T denote the resulting PQR-tree.
   (3) Assign a number K_v for each node v of T_N such that x is the parent of y if and only if $K_x < K_y$.
   (4) N←N+1.
3. Let S be the set of those leaves of T corresponding to the vertex N of G. **If** a reduction of T for S is executable **then goto** Step 5.
4. .Find a minimum set of leaves of T such that, after deleting edges incident upon those leaves from T, we can execute a reduction of the resulting PQR-tree for S. {Note that there is one-to one correspondence between such edges and leaves. This procedure will be explained in the following subsection.}
5. In the PQR-tree obtained after one reduction of Step 4, coalesce all elements of S, appearing consecutively, into one leaf (this leaf corresponds to the vertex N), and let T denote the resulting PQR-tree. **If** N=n **then** halt **else goto** Step 2.

3.2. Explanation of Step 4.

We explain the details of Step 4 in PLAN-PWB: how to find a minimum set of edges whose deletion from the current PQR-tree enable at least one reduction of the resulting PQR-tree. Before describing the outline of the algorithm of Step 4, we need a few more definitions.

Let T be a PQR-tree and S be a leaf set of T. The *S-tree* (a counterpart of the pertinent subtree in [9]) of T is the PQR-subtree of T having the smallest height among those T' with S⊆F(T'). The root and and those leaves contained in S of the S-tree are called the *S-root* and *S-leaves*, respectively. A node of T is called an *S-node* if an S-leaf is one of its descendants. We define five types B, W, HL, HR and A of a node v which is a P-node, a Q-node or an R-node contained in the S-tree as follows.

B: all leaves that are descendants of v are S-leaves.

W: no leaves that are descendants of v are S-leaves.

HL (HR, respectively):S-leaves appear consecutively from the leftmost (the

rightmost) of the frontier consisting of all leaves that are descendants of v.

A: S-leaves appear consecutively in the middle with non-S-leaves running to both the left and the right in the frontier consisting of all leaves that are descendant of v.

B, W and A are the same as those in [13], while H of [13] is divided into HL and HR to handle R-nodes. Any leaf is either B or W. For each S-node v, let b_v, w_v, hL_v, hR_v or a_v denote the minimum of leaves whose deletion make that S-node B, W, HL, HR or A, respectively. In the computation of those numbers, B or W are considered as a special case of each of HL, HR and A. Let L_v denote the total number of descendants of a node v.

Now we describe the outline of Step 4 consisting of six steps as follows.

Step4-1. Compute the total number c'_v of children of each node v, by beginning with leaves and proceeding from children to their parents in the current PQR-tree T.

Step4-2. Compute b_v of each non-S-node by means of the procedure PRE_SCAN described in the following subsection.

Step4-3. For each S-node v, compute the total number c_v of S-nodes that are children of v, by beginning with the S-nodes and proceeding from children to their parents in the PQR-tree T. Determine the S-root by comparing numbers K_v.

Step4-4. For each S-node v, compute b_v, w_v, hL_v, hR_v or a_v, from those values already obtained for children of v, by means of the procedure SCAN described in the following subsection.

Step4-5. (1)Determine the type of the S-root r as the one with the minimum among the values b_r, w_r, hL_r, hR_r and a_r. Accordingly determine the type of each S-node and assign the corresponding label B, W, HL, HR or A, by proceeding from parents to their children.
(2)For each node, delete nodes or leaves as specified by the label and the values assigned. This is also done from the root in the breadth-first manner. (The details will be explained in the following subsection.)

Step4-6. Execute a reduction of the current T for S. {At this stage there exists at least one executable reduction.}

3.3. Minimum Deretion of Leaves.

There are two procedures PRE_SCAN and SCAN. PRE_SCAN computes the value b_u of each non-S-node u, and SCAN does b_v, w_v, hL_v, hR_v and a_v of each S-node v.

3.3.1. Computing the number of leaves L_v. When T_N is created in Step 2 of PLAN-PWB, the number of leaves of T_N is equal to the degree $dG(N)$ of N in G. Note that the root of T_N is not always a P-node. This differs from the discussion of PQ-trees. After T_N is added to T by coalescing the root of T_N into the leaf representing the vertex N, set $L_v \leftarrow L_v + dG(N)$ for each ancestor v of the leaf in the resulting tree.

3.3.2. Computing b_u of a non-S-node u. A non-S-node u has $w_u=0$ and therefore, $hL_u=hR_u=a_u=0$. If u is a leaf then

$$b_u=\begin{cases} 1 & \text{if } e_u \in E_0 \\ |E|+1 & \text{if } e_u \in E_1 \cup E_2, \end{cases}$$

where c_u is the edge of G represented by u in PQR-trees. The setting $b_u = |E|+1$ presents the deletion of edges representing modules.

<PRE_SCAN>

1. Put all non-S-leaves into a queue, and set
$$b_u=\begin{cases} 1 & \text{if } e_u \in E_0 \\ |E|+1 & \text{if } e_u \in E_1 \cup E_2, \end{cases}$$
for each leaf u that is a non-S-node.
2. Take the top X out of the queue. **If** the queue is empty or X is the root of T **then** halt.
3. **If** X is an S-node **then goto** Step 4 **else** $bP(X) \leftarrow bP(X)+bX$, where P(X) is the parent of X.
4. $c'P(X) \leftarrow c'P(X)-1$. **If** $c'P(X) \geq 1$ **then goto** Step 2 **else** {$c'P(X)=0$} put P(X) into the queue {P(X) is the rear} and **goto** Step 2.

3.3.3. Computing b_v, w_v, hL_v, hR_v and a_v of an S-node v. We compute b_v, w_v, hL_v, hR_v and a_v of S-node v, by beginning with leaves and processing from children to parents. We provide a queue and a stack for storing nodes of PQR-trees. That queue is used in computing those values by proceeding from leaves to the root, while the stack helps the labeling process which proceeds from the root to leaves.

<SCAN>

1. Put all S-leaves into both the queue and the stack. For each S-leaf, set
$$w_v=\begin{cases} 1 & \text{if } e_v \in E_0 \\ |E|+1 & \text{if } e_v \in E_1 \cup E_2. \end{cases}$$
2. Take the top X out of the queue and compute hLX, hRX and a_X by means of the formulas given in Theorems 2 through 4.
3. **If** X is an S-root **then** halt **else** $b_{P(X)} \leftarrow b_{P(X)}+bX$ and $w_{P(X)} \leftarrow w_{P(X)}+w_X$.
4. $c_{P(X)} \leftarrow c_{P(X)}-1$. **If** $c_{P(X)} \geq 1$ **then goto** Step 2 **else** {$c_{P(X)}=0$} put P(X) into both the queue and the stack, and **goto** Step 2.

We show some theorems in which formulas to compute hLX, hRX and aX are given. Let $1,2,\cdots,m$ be the children of X.

Theorem 1[13]. If x is leaf then $hLX=hRX=a_X=0$.

We omit the subscript X for simplicity in the following theorems.

Theorem 2. if X is P-node then

$$h_L=\sum_{i=1}^{m} \min\{w_i,b_i\}-\max_{1\le i\le m}[\min\{w_i,b_i\}-h_{Li}] \qquad (1),$$

$$h_R=\sum_{i=1}^{m}\min\{w_i,b_i\}-\max_{1\le i\le m}[\min\{w_i,b_i\}-h_{Ri}] \quad (2),$$

$$a=\min\{a_1,a_2\} \quad (3),$$

where

$$a_1=\sum_{i=1}^{m}\min\{w_i,b_i\}$$

$$-\max_{1\le i\ne j\le m}[\min\{w_i,b_i\}-h_{Li}+\min\{w_j,b_j\}-h_{Rj}] \quad (4),$$

$$a_2=\sum_{i=1}^{m}w_i-\max_{1\le i\le m}(w_i-a_i) \quad (5).$$

Proof. We consider only HL, since HR can be handled symmetrically. X is HL if and only if one child of X is HL and the others are either B or W. Choose a child k such that $\min\{w_i,b_i\}-h_{Li}$ is maximum among children $1\le i\le m$, and make k HL. For any other child $j(\ne k)$, make j B if $b_j\le w_j$, or W if $b_j>w_j$. This choice minimizes the number of leaves to be deleted, and (1) follows.

There are two ways of making X A. The first one makes one child and another one HL and HR, respectively, and the others are made either B or W. In this case (4) follows similarly to (1) or (2). The second one makes one child A and the others are made W. In this case choose a child k such that w_i-a_i is maximum among children $1\le i\le m$, and make it A. Then the number of leaves to be deleted is minimized, and (5) follows. Taking the minimum of (4) and (5) gives the formula (3). Q. E. D.

Theorem 3. If X is a Q-node then

$$h_L=\min_{1\le k\le m}\left[\min\left\{\sum_{i=1}^{k-1}(w_i-b_i)-b_k+\sum_{i=1}^{m}b_i,\ \sum_{i=1}^{k-1}(b_i-w_i)-w_k+\sum_{i=1}^{m}w_i\right\}+h_{Lk}\right] \quad (6),$$

$$h_R=\min_{1\le k\le m}\left[\min\left\{\sum_{i=1}^{k1}(w_i-b_i)-b_k+\sum_{i=1}^{m}b_i,\ \sum_{i=1}^{k1}(b_i-w_i)-w_k+\sum_{i=1}^{m}w_i\right\}+h_{Rk}\right] \quad (7),$$

$$a=\min\{a_1,a_2\} \quad (8),$$

where

$$a_1=\sum_{i=1}^{m}b_i+\min\left[\min_{1\le j\ne k\le m}\left\{\sum_{i=1}^{j-1}(w_i-b_i)+h_{Lj}-b_j+\sum_{i=k+1}^{m}(w_i-b_i)+h_{Rk}-b_k\right\},\ \min_{1\le j<k\le m}\left\{\sum_{i=1}^{j-1}(w_i-b_i)+h_{Rj}-b_j+\sum_{i=k+1}^{m}(w_i-b_i)+h_{Lk}-b_k\right\}\right] \quad (9),$$

$$a_2=\sum_{i=1}^{m}w_i+\min_{1\le i\le m}(a_i-w_i) \quad (10).$$

Proof. We again consider only HL. The discussion for HR is symmetric. X is HL if and only if one child is HL, every child on the left is either B or W, and every one on the right is either W or B, respectively. The number of leaves to be deleted so that X many be HL is given by

$$\sum_{i=1}^{k1}w_i+h_{Lk}+\sum_{i=k+1}^{m}b_i \quad \text{or} \quad \sum_{i=k+1}^{m}b_i+h_{Lk}+\sum_{i=1}^{k1}w_i \quad (11),$$

respectively. Hence choosing k which minimizes (11) gives

$$\min_{1\le k\le m}\left[\min\left\{\sum_{i=1}^{k1}w_i+\sum_{i=k+1}^{m}b_i,\ \sum_{i=1}^{k1}b_i+\sum_{i=k+1}^{m}w_i\right\}+h_{Lk}\right] \quad (12),$$

and (6) follows.

There are two ways of making X A. The first one makes one child j and another one k HL and HR, respectively, every child between them B, and the others W. The number of leaves to be deleted in this case is given by (13-a) or (13-b):

$$\sum_{i=1}^{j-1}w_i+h_{Lj}+\sum_{i=j+1}^{k1}b_i+h_{Rk}+\sum_{i=k+1}^{m}w_i$$
$$=\sum_{i=1}^{j-1}w_i+h_{Lj}+\sum_{i=1}^{m}b_i-\sum_{i=1}^{j}b_i-\sum_{i=k}^{m}b_i+h_{Rk}+\sum_{i=k+1}^{m}w_i$$
$$=\sum_{i=1}^{m}b_i+\sum_{i=1}^{j-1}(w_i-b_i)+h_{Lj}-b_j+\sum_{i=K+1}^{m}(w_i-b_i)+h_{Rk}-b_k \quad (13\text{-a}),$$

$$\sum_{i=1}^{j-1}w_i+h_{Rj}+\sum_{i=j+1}^{k1}b_i+h_{Lk}+\sum_{i=k+1}^{m}w_i$$
$$=\sum_{i=1}^{j-1}w_i+h_{Rj}+\sum_{i=1}^{m}b_i-\sum_{i=1}^{j}b_i-\sum_{i=k}^{m}b_i+h_{Lk}+\sum_{i=k+1}^{m}w_i$$
$$=\sum_{i=1}^{m}b_i+\sum_{i=1}^{j-1}(w_i-b_i)+h_{Rj}-b_j+\sum_{i=K+1}^{m}(w_i-b_i)+h_{Lk}-b_k \quad (13\text{-b}).$$

Hence (9) follows The second one makes one child A and the others W. The number of leaves to be deleted in this case is

$$\sum_{i=1}^{j-1}w_i+a_j+\sum_{i=j+1}^{m}w_i=\sum_{i=1}^{m}w_i-w_j+a_j \quad (14),$$

and (10) follows. Taking the minimum of (9) and (10) gives the formula (8). Q. E. D.

Theorem 4. If X is an R-node then

$$h_L=\min_{1\le k\le m}\left[\sum_{i=1}^{k1}(b_i-w_i)-w_k+\sum_{i=1}^{m}w_i+h_{Lk}\right] \quad (15),$$

$$h_R=\min_{1\le k\le m}\left[\sum_{i=1}^{k1}(w_i-b_i)-b_k+\sum_{i=1}^{m}b_i+h_{Rk}\right] \quad (16),$$

$$a=\min\{a_1,a_2\} \quad (17),$$

where

$$a_1=\sum_{i=1}^{m}b_i+\min_{1\le j<k\le m}\left[\sum_{i=1}^{j-1}(w_i-b_i)+h_{Rj}-b_j+\sum_{i=K+1}^{m}(w_i-b_i)+h_{Lk}-b_k\right] \quad (18),$$

$$a_2=\sum_{i=1}^{m}w_i+\min_{1\le i\le m}(a_i-w_i) \quad (19).$$

Proof. We consider only HL. X is HL if and

only if one child is HL, every child on the left is B and every one on the right is W. Then (15) follows similarly to (6) for a Q-node in Theorem 3.

There are two ways of making X A. The first one makes one child and another one HL and HR, reprectively, every one between them B and the others W, where the child of HR has to be located to the left of that of HL. Then (18) follows similarly to (9) for a Q-node in Theorem 3. The second one makes one child A and the others W. This is the same as the case of a Q-node, and (19) follows. Taking the minimum of (18) and (19) gives (17). Q.E.D.

Scanning children 1 through m, once for each, is enough for us to compute (1), (2), (4), (5), (6), (7), (10), (15), (16), and (19). We can compute (18) by a bidirectional scanning: 1 through m and reversely m through 1, once for each. (9) can be computed by repeating this bidirectional scanning twice.

If deletion of S-nodes and that of non-S-nodes result in the same total number of leaves to be deleted then we decide the types of nodes so that non-S-nodes are forced to be deleted. This is because, as PLAN-PWB proceeds to processing a vertex with larger st-numbering, a smaller degree of that vertex likely reduces the number of leaves to be deleted.

3.4. Labeling and Deletion

Suppose that b_r, w_r, hL_r, hR_r and a_r for the S-root r have been computed in Step 4-4 of PLAN-PWB and that the type of r has been determined by choosing the minimum among them. We repeat three procedures: taking a node v out of the stack, determining the type of that node and assigning the corresponding label to it. The children to be deleted are determined according to the label of v. If such children exist then delete them. If they are leaves then they are put into a list, and if not, they are assigned the label D. If a node assigned D is a non-S-node then it is put into another queue. A node assigned D will be taken out of the stack (S-nodes) or this second queue (non-S-nodes) later, and processing the children is repeated as above.

3.5. Time Complexity

The most time-consuming pat of PLAN-PWB is the computation of the minimum number of leaves to be deleted. This part takes time proportional to the number of nodes in a PQR-tree T, and at most $O(|V|+|E|)$ nodes exist in any PQR-tree. Since there is $O(|V|)$ repetition, time complexity of PLAN-PWB is $O(|V|(|V|+|E|))$.

4. Experimental Results

The Proposed algorithm PLAN-PWB and REPEAT-PLAN [14], both having $O(|V||E|)$ time complexity, are implemented on a workstation NEC EWS-4800 by using C language. Table 1 shows some of our experimental results(in complete connection for multi-terminals), from which it is recognized that PLAN-PWB produces a smaller set of jumpers than REPEAT-PLAN and that the computation time in millisecond of the first one is also less than that of the second one. Fig. 4. shows the circuit corresponding to data 1, where three jumpers given by PLAN_PWB are denoted by broken lines in (a), while five jumpers given by REPEAT-PLAN are shown by broken lines in (b).

5. Concluding Remarks

An $O(|V||E|)$ algorithm PLAN-PWB for finding minimal set of jumpers in the design of printed wiring boards of analog circuits is proposed. A providing interactive tools useful to designers and improving graphics are under development.

6. Acknowledgement

The authors would like to thank. Eiki Kida, the Department General Manager, Computer Aided Design Center, Audio Systems Research Laboratories, Audio Systems Group, SHARP Corporation, for his suggestion on CAD of VLSI. The research of T. Watanabe is partly supported by the Telecommunication, Advancement Foundation (TAF). This work was partly supported by the Grand in Aid for Scientific Research of the Ministry of Education, Science and Culture of Japan under Grand: (C) 01550289.

Reference

[1]Vanlier, M.C. and Otten, R.H. :"On the mathematical formulation of the wiring problem", Int.J.Circuit theory & Appl., 1, pp.137-147(1973)

[2]Goldstein, A.J. and Schweikert ,D.G. :"A proper model for testing the planarity of electrical circuits", Bell Syst. Tech. J., 52, pp.135-142(1973)

[3]Uehara, T.,Takahashi, O. and Shiraishi, H. :"Program of automatic circuit drawing for mask making system", FUJITSU Scientific & Technical Journal, 8, 3, pp.35-58(1972)

[4]Shirakawa, I.:"Applications of Graph Theory to Packing Design", IECE, vol. 62, July, pp.780-789(1979-07)

[5]Fisher, G.J., Wing, Q. :"Computer recognition and extraction of planer Graphs from the incidence matrix", IEEE Trans. CT-13 p.154(1966)

[6]Nicolson, T.A.J. :"Permutation procedure for minimizing the number of crossings in a network", Proc. Inst. Elect. Engrs, 115, 1, p.21(1968)

[7]Chiba, T., Nishioka, I. and Shirakawa, I.:"An algorithm of maximal planarization of graphs", Proc. 1979 ISCA P.649(1979-07)

[8]Lempel, A., Even, S. and Cederbaum, I.:"An algorithm for planarity testing of graphs", Theory of graphs:International Symposium: Rome, July, 1966(P.Rosenstiehl, Ed.) pp.215-232, Gordon and breath, New york, 1967

[9]Booth,K.S. and Lueker,G.S:"Testing for the

consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms",J. Comput. & Syst. Sci., 13, pp.335-379

[10]Even, S. and Tarjan, R.E. :"Computing an st-numbering", Theoretical Computer Science, 2, pp.339-344(1976)

[11]Aho, A.V., Hopcroft, J.E. and Ullman, J.D.:"The design and analysis of computer algorithms", Addison-Wesley, Reading(1974).

[13]Ozawa, T. and Takahashi, H.:"An Algorithm for Planarization of Graphs Using a PQ-Tree", Technical Research Report CAS79-150 IEICE of Japan ,pp.25-30(1979).

[14]Masuda, S., Kashiwabara, T. and Fujisawa, T.:"A Wiring Problem on Single Layer Printed Circuit Board without Mounting Modules Upside-Down", IEICE Trans. Vol. J66-A NO.3, pp.235-242(1983).

[15]Jayakumar, R., Thulasiraman, K., and Swamy, M.N.S. :"$O(n^2)$ Algorithms for Graph Planarization", IEEE Trans. Computer-Aided Design, Vol. 8, No. 3, March 1989.

[16]Ozawa, T. and Takahashi, H. :"A graph-pla narization algorithm and its application to random graphs", in Graph Theory and Algorithms, Springer-verlag Lecture Notes in Computer Science, Vol. 108, pp. 95-107, 1981

[17]Araki, T., Iwamoto, K., Watanabe, T., and Onaga, K.:"Finding a Minimal Set of Jumpers in the Design of Printed Wiring Boards for Analog Circuits", Technical Research Reports IPS of Japan(May 1990), to appear.

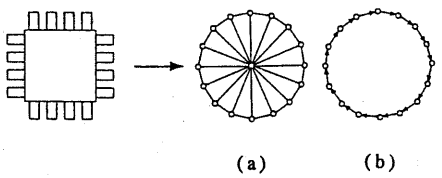


Fig. 2.1. Graph representations of modules.

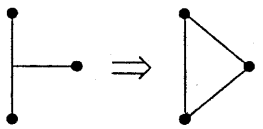


Fig. 2.2. A star-shaped connection and its net representation.

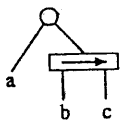
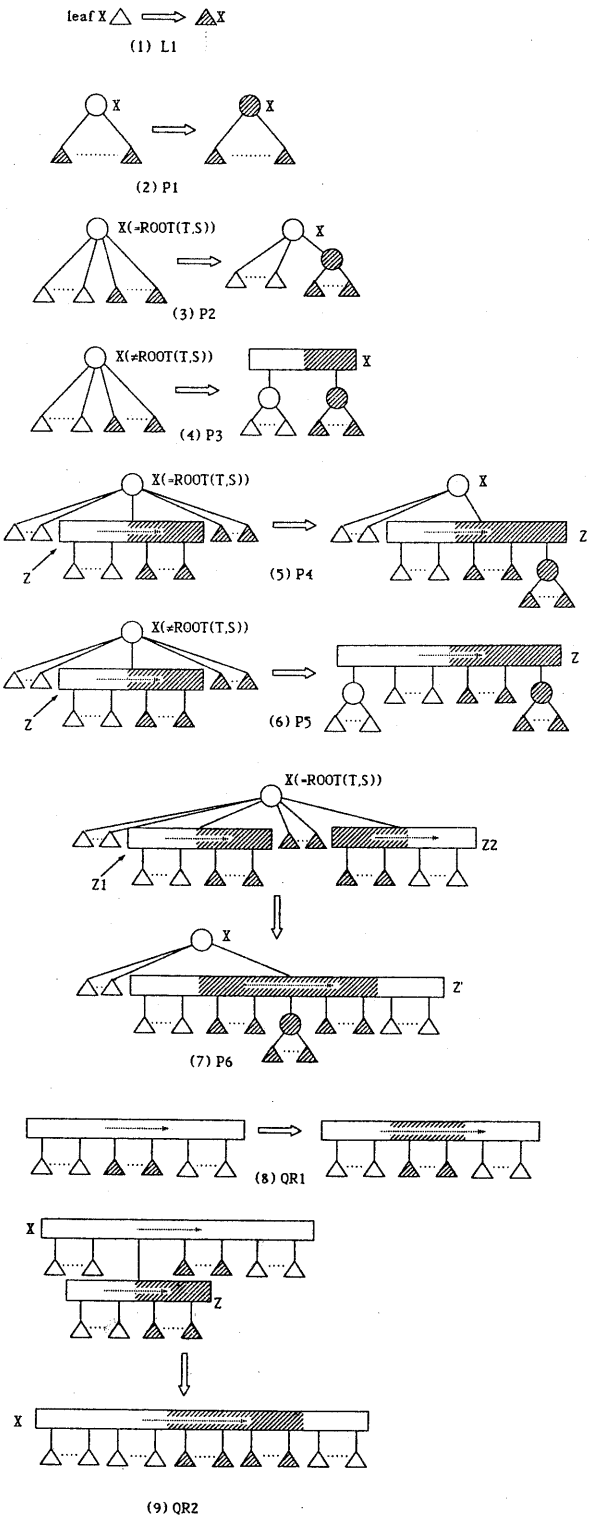


Fig. 2.3. An example of a PQR-tree.



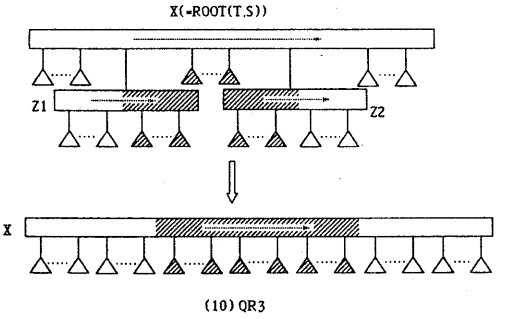


Fig. 2.4. Template matchings for reductions of PQR-trees.

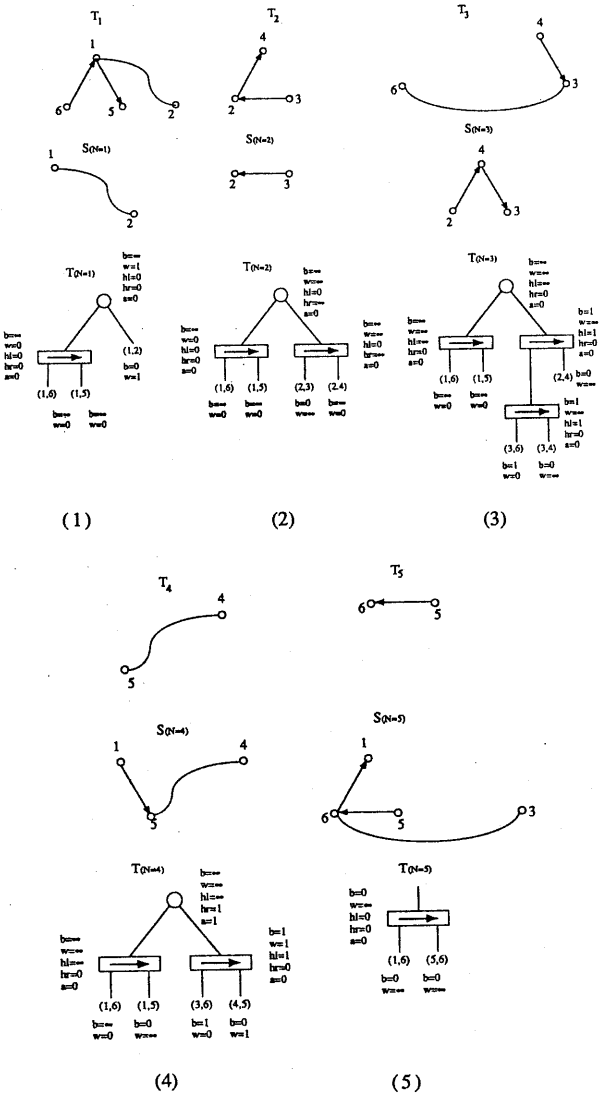


Fig. 3.2. Reductions of PQR-trees for the graph G of Fig. 3.1.

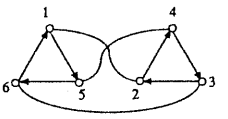


Fig. 3.1. A graph G

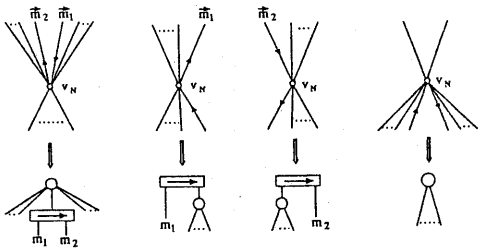


Fig. 3.3. Construction of a PQR-tree TN.

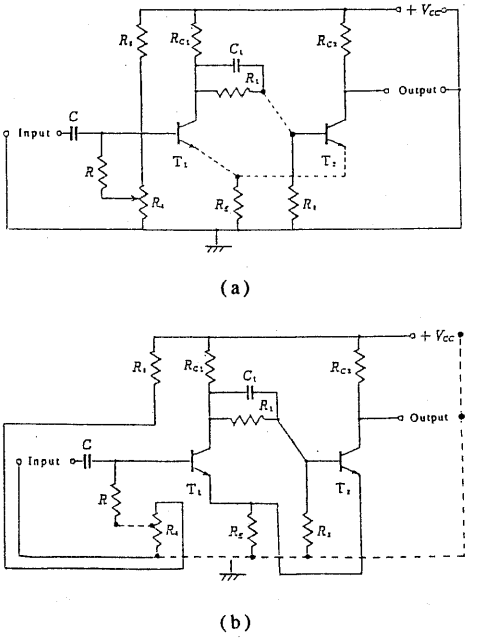


Fig. 4. The circuit corresponding data1, where jumpers are denoted by broken lines : (a) by PLAN-PWB, (b) by REPEAT-PLAN.

DATA#	|V|	|E|	# Jumper		Time (ms)	
			PLAN-PWB	REPEAT-PLAN	PLAN-PWB	REPEAT-PLAN
1	33	45	14	16	11.199	141.760
2	33	48	11	13	58.281	161.643
3	19	21	2	3	3.455	23.182
4	32	40	10	7	7.816	127.544
5	36	39	7	8	7.033	164.250
6	77	197	85	87	23.102	139.988

Table 1. Some experimental results on PRIDE. Time is in millisecond.