SRC TR 89-20

Circular Permutation Layout with Prescribed Between-Pin Congestions

by

C. S. Rim, S. Masuda

and

K. Nakajima

Circular Permutation Layout with Prescribed Between–Pin Congestions [1]

Chong S. Rim
Electrical Engineering Department and Systems Research Center
University of Maryland, College Park, MD 20742

Sumio Masuda
Department of Information and Computer Sciences
Osaka University, Toyonaka, Osaka 560, Japan

Kazuo Nakajima
Electrical Engineering Department,
Institute for Advanced Computer Studies and Systems Research Center
University of Maryland, College Park, MD 20742

[1] This work was supported in part by National Science Foundation grants MIP-84-51510 and CDR-88-03012 (Engineering Research Centers Program), and a grant from AT&T.

Circular Permutation Layout with Prescribed Between–Pin Congestions

by

Chong S. Rim, Sumio Masuda and Kazuo Nakajima

Abstract

Suppose that two sets of terminals $t_1, t_2, \ldots, t_n$ and $b_1, b_2, \ldots, b_n$ are located on two concentric circles C_{out} and C_{in}, respectively. Given a permutation π of integers $1, 2, \ldots, n$, the circular permutation layout problem is the problem of connecting each pair of terminals t_i and $b_{\pi(i)}$ for $i = 1, 2, \ldots, n$ with zero width wires in such a way that no two wires that correspond to different terminal pairs intersect each other. In this paper, we present a linear time algorithm for the following case: (i) no wire can cross C_{out}, (ii) a wire can cross C_{in} at most once, and (iii) the number of wires which can pass between each pair of adjacent terminals on C_{in} is prespecified.

1. Introduction

Suppose that two sets of terminals $t_1, t_2, \ldots t_n$ and $b_1, b_2, \ldots, b_n$ are located on two concentric circles C_{out} and C_{in}, respectively. We assume that the circle C_{out} is outside the circle C_{in} and that the terminals on each circle are labeled in ascending order of their subscripts in the clockwise direction. Given a permutation π of integers $1, 2, \ldots, n$, the *circular permutation layout* (CPL) problem is the problem of connecting each pair of terminals t_i and $b_{\pi(i)}$ for $i = 1, 2, \ldots, n$ with zero width wires in such a way that no two wires that correspond to different terminal pairs intersect each other.

The CPL problem was first proposed by Ozawa [7] as an extension of the *linear permutation layout* (LPL) problem which has widely been studied [2,3,4,5,9,10]. As pointed out by Ozawa [7], the CPL problem may arise in the design of hybrid integrated circuits and printed circuit boards (PCBs). Consider, for example, the problem of connecting a series of incoming wires to the pins on a particular module on a PCB (see Fig. 1). We assume that no two pins on the module are electrically equivalent and hence any two wires for the connections must not intersect each other. The order of the incoming wires is previously determined and may not be the same as that of the pins on the module. If only one layer is available to realize the connections, this problem is equivalent to the CPL problem. The terminals on C_{in} correspond to the pins on the module and the terminals on C_{out} represent the incoming wires. Since the pins have fixed spacing and the wires have finite width, we may assume that the number of wires which can pass between two adjacent pins is limited to some

finite integer. No wires may cross the boundary represented by C_{out} since they would intersect the existing incoming wires.

From the observations mentioned above, we consider the CPL problem with the following constraints on the wires in its solution layout:

1. No wire can cross C_{out},
2. A wire can cross C_{in} at most once, and
3. At most $\kappa_i \geq 1$ wires can pass between two adjacent terminals b_i and b_{i+1} on C_{in} for $i = 1, 2, \ldots, n$, with $b_{n+1} = b_1$.

We call a layout in which wires satisfy the above constraints a *CPL–solution.* For example, Fig. 2 shows a CPL–solution for the permutation π = (30 29 26 25 24 16 15 14 9 8 7 13 12 10 11 6 18 17 21 20 23 22 19 5 4 3 28 27 31 2 1 32), where we assume that $\kappa_i = 2$ for all $i = 1, 2, \ldots, 32$. For simplicity, we label each terminal t_i or b_i as i in the figures throughout this paper. Ozawa [7] developed an $O(n^2)$ time algorithm for finding a CPL–solution if one exists for the special case of $\kappa_i = 1$ for all $i = 1, 2, \ldots, n$. Recently, Rim *et al.* [8] developed a linear time algorithm for the same special case.

In this paper, we present an $O(n)$ time algorithm for finding a CPL–solution for the more general case described above. In Section 2 we introduce some basic definitions and notation. Section 3 provides some useful properties of a CPL–solution. In Section 4 we explain merging operations, which play an important role in our algorithm. The algorithm is presented in Section 5. Quite recently, another linear time algorithm for the same case as ours was independently discovered [6].

2. Definitions and Notation

Let π be a given permutation of integers $1, 2, \ldots, n$. If there exists a CPL–solution for π, we say that π is *realizable*. A *net* n_i is an ordered pair of terminals $(t_i, b_{\pi(i)})$ which must be electrically connected. Let w_i denote the wire which connects the terminals of n_i. If w_i crosses the circle C_{in}, it is called an *indirect wire*; otherwise it is called a *direct wire*. For example, in Fig. 2, w_{31} is a direct wire and w_1 is an indirect wire. We assume that a net is routed by an indirect wire if and only if it can not be replaced by a direct wire. We define $l \oplus 1$ to be $l + 1$ if $1 \leq l \leq n - 1$ and 1 if $l = n$. If a wire w_i crosses C_{in} between two terminals b_j and $b_{j\oplus 1}$, we denote this fact by $w_i \wr \bullet b_{j\oplus 1}$. For example, $w_1 \wr \bullet b_3$ in Fig. 2.

Let $N(\pi) = \{n_i = (t_i, b_{\pi(i)}) \mid 1 \leq i \leq n\}$ be the set of nets. Let $p_1, p_2, \ldots, p_k$ be distinct integers between 1 and n. A sequence $[p_1, p_2, \ldots, p_k]$ is called an *increasing consecutive sequence* (**icseq**) if $p_i \oplus 1 = p_{i+1}$ for $i = 1, 2, \ldots, k - 1$. Let $M = \{n_{p_i} = (t_{p_i}, b_{q_j}) \mid 1 \leq i, j \leq m\}$ be a nonempty subset of $N(\pi)$ such that $[p_1, p_2, \ldots, p_m]$ and $[q_1, q_2, \ldots, q_m]$ are **icseq**s. We say that M is *locally realizable* if the nets in M can be routed in such a way that the wires satisfy the three constraints on a CPL–solution and the constraint that every wire w_{p_i}, $i = 1, 2, \ldots, m$, is either a direct wire or an indirect wire which passes between two adjacent terminals in $\{b_{q_0}, b_{q_1}, \ldots, b_{q_m}, b_{q_m\oplus 1}\}$, where q_0 is an integer such that $q_1 = q_0 \oplus 1$. A *local realization* of M is a layout of M in which wires satisfy all such constraints. In particular, the fourth constraint will be referred to as the condition of local realizability. For example, the layout of $D = \{n_{17}, n_{18}, n_{19}, n_{20}, n_{21}, n_{22}, n_{23}\}$ in Fig. 2 satisfies all the constraints, and hence D

is locally realizable. Suppose that M is locally realizable and let L_M be any local realization of M. If there is a wire w in L_M such that $w \wr \bullet b_{q_1}$ (resp., $w \wr \bullet b_{q_{m+1}}$), it is called a *left* (resp., *right*) *boundary wire* of L_M. For example, w_{17} is a left boundary wire and w_{23} is a right boundary wire of the local realization of D in Fig. 2. Note that if $M = N(\pi)$, both the left and right boundary wires of $L_{N(\pi)}$ pass between the same two adjacent terminals.

Let $M' = \{n_{p'_i} = (t_{p'_i}, b_{q'_j}) \mid 1 \leq i, j \leq l\}$ be another subset of $N(\pi)$ such that $[p'_1, p'_2, \ldots, p'_l]$ and $[q'_1, q'_2, \ldots, q'_l]$ are **icseqs**. We say that M is *parallel* to M', denoted by $M \mid\mid M'$, if and only if $p_m \oplus 1 = p'_1$ and $q_m \oplus 1 = q'_1$. For example, $\{n_{29}\} \mid\mid \{n_{30}, n_{31}, n_{32}\}$ and $\{n_{19}, n_{20}\} \mid\mid \{n_{21}, n_{22}\}$ in Fig. 2. We now introduce clusters which play an important role in our algorithm for finding a CPL–solution. We call M a *cluster* in $N(\pi)$ if (i) $M \neq N(\pi)$ and $\pi(p_i) = q_{k-i+1}$ for $i = 1, 2, \ldots, k$, or (ii) $M = N(\pi)$ and there is an integer x such that $1 \leq x \leq n$ and $\pi(p_i) = q_{(n-i)\oplus x}$ for $i = 1, 2, \ldots, n$. If a cluster consists of only one net, it is called a *trivial cluster*; otherwise it is called a *nontrivial cluster*. A cluster is *maximal* if and only if it is not contained in any other cluster. For example, the instance shown in Fig. 2 has the following sixteen maximal clusters: $\{n_1, n_2\}$, $\{n_3, n_4, n_5\}$, $\{n_6, n_7, n_8\}$, $\{n_9, n_{10}, n_{11}\}$, $\{n_{12}, n_{13}\}$, $\{n_{14}\}$, $\{n_{15}\}$, $\{n_{16}\}$, $\{n_{17}, n_{18}\}$, $\{n_{19}, n_{20}\}$, $\{n_{21}, n_{22}\}$, $\{n_{23}\}$, $\{n_{24}, n_{25}, n_{26}\}$, $\{n_{27}, n_{28}\}$, $\{n_{29}\}$ and $\{n_{30}, n_{31}, n_{32}\}$. It is easy to see that if two distinct clusters C and C' are both maximal, $C \cap C' = \phi$. Furthermore, if $C \cup C' = N(\pi)$, $N(\pi)$ itself is a cluster. Thus, if $N(\pi)$ is not a single cluster, it can uniquely be partitioned into three or more maximal clusters.

3. Layout of Maximal Clusters

Assume that $N(\pi)$ consists of three or more maximal clusters. Let $C = \{n_{u_i} = (t_{u_i}, b_{v_{k-i+1}}) \mid 1 \leq i \leq k\}$ be a cluster in $N(\pi)$, where $[u_1, u_2, \ldots, u_k]$ and $[v_1, v_2, \ldots, v_k]$ are **icseqs**. Let $\lceil x \rceil$ denote the smallest integer that is not less than x.

Lemma 1. *C is locally realizable.*

Proof. Since $\kappa_i \geq 1$ for $i = 1, 2, \ldots, n$, the nets in C can always be realized by one of the following two types of layouts:

1. A Type DL layout of C is the layout such that $w_{u_{\lceil (k+1)/2 \rceil}}$ is a direct wire and $w_{u_i} \wr \bullet b_{v_i}$ for all $i = 1, 2, \ldots, k$ with $i \neq \lceil (k+1)/2 \rceil$ (see Fig. 3 (a)).
2. A Type DR layout of C is the layout such that $w_{u_{\lceil k/2 \rceil}}$ is a direct wire and $w_{u_i} \wr \bullet b_{v_i \oplus 1}$ for all $i = 1, 2, \ldots, k$ with $i \neq \lceil k/2 \rceil$ (see Fig 3 (b)). □

Let $M = \{n_{p_i} = (t_{p_i}, b_{q_j}) \mid 1 \leq i, j \leq m\}$ be a nonempty subset of $N(\pi)$ such that $[p_1, p_2, \ldots, p_m]$ and $[q_1, q_2, \ldots, q_m]$ are **icseqs**. Assume that M is locally realizable and let L_M be any local realization of M. For the following two lemmas, we assume that M consists of one or more maximal clusters. Let $C = \{n_{u_i} = (t_{u_i}, b_{v_{k-i+1}}) \mid 1 \leq i \leq k \leq m\}$ be one such cluster in M, where $[u_1, u_2, \ldots, u_k]$ and $[v_1, v_2, \ldots, v_k]$ are **icseqs**. Let L_C denote the layout of C in L_M. It is easy to see that L_C has at most one direct wire.

Lemma 2. *Suppose that C is a maximal cluster. L_C satisfies the condition of local realizability if and only if it has a direct wire.*

Proof. It is easy to see that if L_C satisfies the condition of local realizability, it has a direct wire. Assume that L_C has a direct wire. If $|C| = 1$ or M itself is a maximal

cluster, L_C is a local realization of C. Assume that $|C| \geq 2$ and M consists of two or more maximal clusters. Suppose that L_C does not satisfy the condition of local realizability. Let $n_{u_x} = (t_{u_x}, b_{v_{k-x+1}})$ be the net that is routed by a direct wire in L_C. We first consider the case in which $x \geq 2$. Let n_{u_y}, $1 \leq y \leq x-1$, be a net whose corresponding wire violates the condition of local realizability. See Fig. 4. Let $n_s = (t_s, b_{\pi(s)})$ be the net in M with $\pi(s) \oplus 1 = v_1$. This net is routed by an indirect wire. It is easy to see that if $u_k \oplus 1 \neq s$, $w_{u_k \oplus 1}$ will violate the second constraint on a CPL–solution. Therefore, $u_k \oplus 1 = s$ and hence $C \cup \{n_s\}$ forms a cluster in M, which contradicts the maximality of C. Similarly, if $x = 1$, we can show that the wires w_{u_2}, $w_{u_3}, \ldots, w_{u_k}$ pass between b_{v_k} and $b_{v_k \oplus 1}$. □

Lemma 3. If C is a nontrivial cluster and L_C in L_M satisfies the condition of local realizability, L_C has at least one boundary wire.

Proof. If w_{u_1} is not a left boundary wire, w_{u_k} can not be a direct wire and it has to pass between b_{v_k} and $b_{v_k \oplus 1}$. Thus, w_{u_k} is a right boundary wire. Similarly, if w_{u_k} is not a right boundary wire, w_{u_1} is a left boundary wire. □

We now assume that M consists of two or more maximal clusters. Furthermore, we assume that there exists at least one indirect wire in L_M. Let $B_M = \{b_{q_1}, b_{q_2}, \ldots, b_{q_m}\}$ and let q_0 and q_{m+1} denote the integers such that $q_1 = q_0 \oplus 1$ and $q_{m+1} = q_m \oplus 1$, respectively. Let $C = \{n_{p_{x+i}} = (t_{p_{x+i}}, b_{q_{y+k-i-1}}) \mid 0 \leq i \leq k-1\}$ be a cluster in M whose nets are all routed by indirect wires in L_M. Assume that $w_{p_{x+k-1}} \wr \bullet b_{q_u}$ and $w_{p_x} \wr \bullet b_{q_{v+1}}$ in L_M for some u and v such that $1 \leq u \leq m+1$ and $0 \leq v \leq m$. Let $B_1 = \{b_{q_u}, b_{q_{u+1}}, \ldots, b_{q_{y-1}}\}$, $T_1 = \{t_{p_{x+k}}, t_{p_{x+k+1}}, \ldots, t_{p_{x+k+y-u-1}}\}$, $B_2 = \{b_{q_{y+k}}, b_{q_{y+k+1}},$

$\ldots, b_{q_v}\}$ and $T_2 = \{t_{p_{x+k+y-v-1}}, t_{p_{x+k+y-v}}, \ldots, t_{p_{x-1}}\}$ such that $[q_u, q_{u+1}, \ldots, q_{y-1}]$, $[p_{x+k}, p_{x+k+1}, \ldots, p_{x+k+y-u-1}]$, $[q_{y+k}, q_{y+k+1}, \ldots, q_v]$ and $[p_{x+k+y-v-1}, p_{x+k+y-v}, \ldots, p_{x-1}]$ are **icseqs**. See Fig. 5. Note that $|B_1| = |T_1|$ and $|B_2| = |T_2|$. We define two sets of terminals B_C and T_C in the following way: $B_C = B_1$ (resp., B_2) and $T_C = T_1$ (resp., T_2) if $B_1 \subset B_M$ (resp., $B_2 \subset B_M$). Note that if $M = N(\pi)$, both B_1 and B_2 are subsets of B_M. In this case, let $B_C = B_1$ (resp., B_2) and $T_C = T_1$ (resp., T_2) if $|B_1| \leq |B_2|$ (resp., if $|B_1| > |B_2|$). Let M_C denote the set of all nets that have terminals in B_C.

Lemma 4. Every net in M_C has a terminal in T_C and M_C is locally realizable.

Proof. Without loss of generality, assume that $B_C = B_1$ and $T_C = T_1$. Also assume that there is a net $n_{p_s} = (t_{p_s}, b_{\pi(p_s)})$ in M_C such that $p_s \notin T_C$. See Fig. 5. Since $|B_C - \{b_{\pi(p_s)}\}| < |T_C|$, there is a wire w which connects a terminal in T_C and a terminal not in B_C. However, the existence of the wire w_{p_s} and the wires for C forces the wire w to cross C_{in} at least twice, which violates the second constraint on a CPL–solution. Therefore, every net in M_C has a terminal in T_C. Since the nets in M_C connect terminals in B_C and those in T_C, it is easy to see that the layout of M_C in L_M satisfies the condition of local realizability, and hence M_C is locally realizable.

□

Lemma 5. M has at least two parallel maximal clusters.

Proof. If the layout of every maximal cluster in M has a direct wire in L_M, it is easy to see that there is a pair of parallel maximal clusters. Assume that there is at least one maximal cluster in M whose nets are all routed by indirect wires in L_M. Let C be a maximal cluster such that $|M_C|$ is a minimum among all such maximal clusters.

Clearly, M_C consists of two or more maximal clusters; otherwise $C \cup M_C$ would form a single cluster. Since $|M_C|$ is a minimum, no maximal cluster in M_C is routed in L_M by indirect wires only. In other words, the layout of every maximal cluster in M_C has a direct wire. Therefore, any two adjacent maximal clusters in M_C are parallel. □

We now assume that π is realizable. Let $L_{N(\pi)}$ be any CPL–solution for π. The following lemma provides a useful property of parallel maximal clusters.

Lemma 6. Let C and C' be maximal clusters in $N(\pi)$. If $C \,||\, C'$, the layouts of C and C' each satisfy the condition of local realizability in $L_{N(\pi)}$.

Proof. Let $C = \{n_{u_i} = (t_{u_i}, b_{v_{r-i+1}}) \mid 1 \leq i \leq r\}$ and $C' = \{n_{u'_i} = (t_{u'_i}, b_{v'_{s-i+1}}) \mid 1 \leq i \leq s\}$ such that $u'_1 = u_r \oplus 1$ and $v'_1 = v_r \oplus 1$. Let $H = \{b_{v_1}, b_{v_2}, \ldots, b_{v_r}\}$. Assume that L_C in $L_{N(\pi)}$ dose not satisfy the condition of local realizability. Then, there is a net, say $n_{p_1} = (t_{p_1}, b_{q_1})$, in C such that w_{p_1} in $L_{N(\pi)}$ passes between terminals b_{q_j} and $b_{q_{j+1}}$ which are not contained in H, where $[p_1, p_2, \ldots, p_n]$ and $[q_1, q_2, \ldots, q_n]$ are **icseqs**. See Fig. 6. Let $B_1 = \{b_{q_{j+1}}, b_{q_{j+2}}, \ldots, b_{q_n}\}$, $T_1 = \{t_{p_2}, t_{p_3}, \ldots, t_{p_{n-j+1}}\}$, $B_2 = \{b_{q_2}, b_{q_3}, \ldots, b_{q_j}\}$ and $T_2 = \{t_{p_{n-j+2}}, t_{p_{n-j+3}}, \ldots, t_{p_n}\}$. Using the similar argument as in the proof of Lemma 4, we can show that every terminal in T_1 (resp., T_2) is connected to a terminal in B_1 (resp., B_2). Since $\{b_{v_1}, b_{v_2}, \ldots, b_{v_{r-x}}\} \subset B_1$, $\{t_{u_{x+1}}, t_{u_{x+2}}, \ldots, t_{u_r}\} \subset T_1$. Since B_1 contains $b_{q_{j+1}}$ which is not in H and $|B_1| = |T_1|$, $t_{u'_1}$ is in T_1. Similarly, since $b_{q_j} \notin H$, $b_{v'_1}$ is in B_2. These two facts imply that all nets that have terminals in B_2 are in either C or C', and hence all the terminals in T_2 belong to those nets. Thus, $N(\pi)$ consists of two maximal clusters, which contradicts our assumption that $N(\pi)$ consists of three or more maximal clusters. Therefore, L_C in $L_{N(\pi)}$ satisfies the condition of local realizability. Similarly, $L_{C'}$ in $L_{N(\pi)}$ satisfies the condition of local

realizability. □

4. Components and Merging Operations

Let $M = \{n_{p_i} = (t_{p_i}, b_{q_j}) \mid 1 \leq i, j \leq m\}$ be a nonempty subset of $N(\pi)$ such that $[p_1, p_2, \ldots, p_m]$ and $[q_1, q_2, \ldots, q_m]$ are **icseqs**. Assume that M consists of one or more maximal clusters. We call M a *component* of $N(\pi)$ if it has a property that if π is realizable, then its layout in any CPL–solution satisfies the condition of local realizability. Locally realizable subsets are not always components. For example, any maximal cluster is locally realizable by Lemma 1, but not all maximal clusters are components (see the cluster $\{n_1, n_2\}$ in Fig. 2). On the other hand, every component is locally realizable as long as π is realizable.

A locally realizable component may have many possible local realizations. In our algorithm, we consider a locally realizable component from the point of view of necessary boundary wires for its local realization. In fact, as will be seen in the description of the algorithm in this and the next sections, only two types of local realization of components are used. Such realizations are specified by three parameters $type(M), l(M)$ and $r(M)$ associated with each component M. They are to be interpreted as follows.

1. If $type(M) =$ 'f', at least $l(M) \geq 0$ left and at least $r(M) \geq 0$ right boundary wires are needed for any local realization of M (see Fig. 7 (a)).
2. If $type(M) =$ 'x', either at least $l(M) \geq 1$ left but no right boundary wires or at least $r(M) \geq 1$ right but no left boundary wires are needed for any local realization of M (see Fig. 7 (b)).

We now assume that a component M is created by our algorithm and $type(M)$ is either 'f' or 'x'. If a component M is realized by a layout with $l(M)$ left and no right (resp., $r(M)$ right and no left) boundary wires, we call such a realization an *l–*realization (resp., an *r–realization*). During the execution of our algorithm, which will be described in the next section, if π is realizable, a component M whose $type(M)$ value is 'x' will be realized as either an *l–*realization or an *r–*realization. The parameter values of M are modified as $type(M) =$ 'f', $l(M) = l(M)$ and $r(M) = 0$ (resp., $type(M) =$ 'f', $l(M) = 0$ and $r(M) = r(M)$) if we realize M as an *l–*realization (resp., an *r–*realization).

A *merging operation* is an operation that creates a new component by combining a component called the *core* with another component or one or two maximal clusters. In our algorithm, $N(\pi)$ is first partitioned into maximal clusters. If a merging operation successfully creates a new component, it also determines the three parameter values of the new component. Then, the algorithm repeatedly finds merging operands and performs a merging operation. If it neither constructs a new component by a merging operation nor finds any merging operands, π is not realizable as will be shown later. On the other hand, if it produces a single component that is locally realizable by merging all the maximal clusters, clearly π is realizable. In the following two subsections, we define two types of merging operations which are used in our algorithm. They are called *P–merging* and *X–merging* operations. In the remaining part of this paper, we assume that any component that is not a single cluster was created by either a P–merging or X–merging operation.

4.1. P–merging Operation

A P–merging operation is to merge two parallel components called *P–merging operands*. Let $M = \{n_{p_i} = (t_{p_i}, b_{q_j}) \mid 1 \leq i, j \leq m\}$ and $M' = \{n_{p'_i} = (t_{p'_i}, b_{q'_j}) \mid 1 \leq i, j \leq l\}$ be components of $N(\pi)$ such that $[p_1, p_2, \ldots, p_m, p'_1, p'_2, \ldots, p'_l]$ and $[q_1, q_2, \ldots, q_m, q'_1, q'_2, \ldots, q'_l]$ are **icseqs**. Assume that both M and M' are locally realizable. It is clear that the combined set $M_{new} = M \cup M'$ is a component of $N(\pi)$. Suppose that $type(M) =$ 'f' and $type(M') =$ 'f'. If $r(M) + l(M') > \kappa_{q_m}$, M_{new} can not be routed without violating the third constraint on a CPL–solution since more than κ_{q_m} wires must pass between the terminals b_{q_m} and $b_{q'_1}$. Therefore, π is not realizable. On the other hand, if $r(M) + l(M') \leq \kappa_{q_m}$, clearly M_{new} is locally realizable. The parameter values of M_{new} will be $type(M_{new}) =$ 'f', $l(M_{new}) = l(M)$ and $r(M_{new}) = r(M')$, which implies that at least $l(M)$ left and at least $r(M')$ right boundary wires are needed for any local realization of M_{new}.

As another example, suppose that $type(M) =$ 'x' and $type(M') =$ 'x'. If $r(M) + l(M') \leq \kappa_{q_m}$, M_{new} is locally realizable with no boundary wires by obtaining an r– and l–realization of M and M', respectively. In this case, the parameter values are set as $type(M_{new}) =$ 'f', $l(M_{new}) = 0$ and $r(M_{new}) = 0$. If $r(M) + l(M') > \kappa_{q_m}$, a local realization of M_{new} will be obtained by using either both l–realizations or both r–realizations of M and M'. In this case, the parameter values are set as $type(M_{new}) =$ 'x', $l(M_{new}) = l(M)$ and $r(M_{new}) = r(M')$.

In Appendix A–1, we describe a complete procedure for a P–merging operation by considering all possible cases. In a P–merging operation, either M or M' can be

the core.

4.2. X–merging Operation

An X–merging operation is to merge a component and one or two maximal clusters. We call such a component and maximal clusters *X–merging operands*. In an X–merging operation, the component is the core. Let $M = \{n_{p_i} = (t_{p_i}, b_{q_j}) \mid 1 \leq i, j \leq m\}$, $C_1 = \{n_{u_i} = (t_{u_i}, b_{v_{r-i+1}}) \mid 1 \leq i, j \leq r\}$ and $C_2 = \{n_{u'_i} = (t_{u'_i}, b_{v'_{s-i+1}}) \mid 1 \leq i, j \leq s\}$ be a component and two maximal clusters, respectively, such that $[u_1, u_2, \ldots, u_r, p_1, p_2, \ldots, p_m, u'_1, u'_2, \ldots, u'_s]$ and $[v'_1, v'_2, \ldots, v'_s, q_1, q_2, \ldots, q_m, v_1, v_2, \ldots, v_r]$ are **icseq**s. Note that if there is no such C_1 (resp., C_2), we set C_1 (resp., C_2) to be an empty set. We define H_1 to be $\{b_{v'_0}, b_{v'_1}, \ldots, b_{v'_s}, b_{q_1}\}$ (resp., $\{b_{q_0}, b_{q_1}\}$) if $C_2 \neq \phi$ (resp., $C_2 = \phi$), where v'_0 and q_0 are integers such that $v'_1 = v'_0 \oplus 1$ and $q_1 = q_0 \oplus 1$, respectively. Similarly, H_2 is defined to be $\{b_{q_m}, b_{v_1}, b_{v_2}, \ldots, b_{v_r}, b_{v_r \oplus 1}\}$ (resp., $\{b_{q_m}, b_{q_m \oplus 1}\}$) if $C_1 \neq \phi$ (resp., $C_1 = \phi$). An X–merging operation is based on the following lemma.

Lemma 7. If π is realizable, every net in C_1 (resp., C_2) is routed by an indirect wire which passes two adjacent terminals in H_1 (resp., H_2) in any CPL–solution for π.

Proof. The component M has at least two maximal clusters; otherwise $C_1 \cup M \cup C_2$ would become a cluster. Similarly, the subset $\bar{M} = N(\pi) - (C_1 \cup M \cup C_2)$ has at least two maximal clusters. Assume that π is realizable and let $L_{N(\pi)}$ be any CPL–solution for π. For any subset D of $N(\pi)$, let L_D denote the layout of D in $L_{N(\pi)}$. Without loss of generality, we only consider the case in which $C_1 \neq \phi$. Assume that a net in C_1 is routed in $L_{N(\pi)}$ by a direct wire or an indirect wire which passes between two

adjacent terminals at most one of which is in H_1. There are three possibilities.

1. A net in C_1 is routed by either a direct wire or an indirect wire which passes between two adjacent terminals in H_2 (see Fig. 8 (a)). By Lemmas 5, 6 and 2, there are at least two direct wires in L_M. Therefore, all nets in $\bar{M}$ must be realized by indirect wires. This implies that $\bar{M}$ consists of a single cluster, not two or more maximal clusters, a contradiction.

2. A net in C_1 is routed by an indirect wire which passes between two adjacent terminals, say b_{q_j} and $b_{q_{j+1}}$ in $\{b_{q_1}, b_{q_2}, \ldots, b_{q_m}\}$ (see Fig. 8 (b)). Clearly, all nets that have terminals in $\{b_{q_1}, b_{q_2}, \ldots, b_{q_j}\}$ must be routed by those indirect wires that violate the condition of local realizability of M, a contradiction.

3. A net in C_1 is routed by an indirect wire which passes between two adjacent terminals not in the set $\{b_{v'_1}, b_{v'_2}, \ldots, b_{v'_s}, b_{q_1}, b_{q_2}, \ldots, b_{q_m}, b_{v_1}, b_{v_2}, \ldots, b_{v_r}\}$ (see Fig. 8 (c)). Let $n_{u_x} \in C_1$ be such a net. Let $n_z = (t_z, b_{\pi(z)})$ be a net in $N(\pi)$ such that $\pi(z) \oplus 1 = v'_1$ (resp., q_1) if $C_2 \neq \phi$ (resp., $C_2 = \phi$). Clearly, w_z is an indirect wire. Let $n_y = (t_y, b_{\pi(y)})$ be a net in $N(\pi)$ such that $y = u'_s \oplus 1$ (resp., $p_m \oplus 1$) if $C_2 \neq \phi$ (resp., $C_2 = \phi$). It is easy to see that if $z \neq y$, w_y in $L_{N(\pi)}$ violates the second constraint on a CPL–solution. Therefore, $z = y$. If $C_2 \neq \phi$, $C_2 \cup \{n_z\}$ would form a cluster, which contradicts the maximality of C_2. If $C_2 = \phi$, the existence of the maximal cluster that contains n_z would contradict the fact that $C_2 = \phi$.

In conclusion, every net in C_1 is routed in $L_{N(\pi)}$ by an indirect wire which passes between two adjacent terminals in H_1. Similarly, every net in C_2 is routed in $L_{N(\pi)}$ by an indirect wire which passes between two adjacent terminals in H_2. □

Lemma 7 implies that if π is realizable, the layout of $M_{new} = C_1 \cup M \cup C_2$ in

any CPL–solution must satisfy the condition of local realizability and hence it is a component. In the remaining part of this subsection, we describe how to test the local realizability of M_{new} and how to set its three parameter values if it is locally realizable.

Case A: Either $C_2 = \phi$ or $C_1 = \phi$.

Without loss of generality, we assume that $C_2 = \phi$. By Lemma 7, every net in C_1 must be routed as a left boundary wire of $M_{new} = C_1 \cup M$. There are two possible cases to consider. Recall that q_0 is the integer such that $q_1 = q_0 \oplus 1$ and $r = |C_1|$.

Case 1. $type(M) = `f'$.

If $r + l(M) > \kappa_{q_0}$, M_{new} is not locally realizable since more than κ_{q_0} wires must pass between b_{q_0} and b_{q_1}. Otherwise, M_{new} is locally realizable and the parameter values are set as $type(M_{new}) = `f'$, $l(M_{new}) = r + l(M)$ and $r(M_{new}) = 0$.

Case 2. $type(M) = `x'$.

If $r > \kappa_{q_0}$, clearly M_{new} is not locally realizable. Otherwise, M_{new} is locally realizable and the parameter values are set as $type(M_{new}) = `f'$, $l(M_{new}) = r$ and $r(M_{new}) = 0$. In this case, an r–realization of M is always chosen.

The case in which $C_1 = \phi$ can be treated in a similar way.

Case B: Both $C_1 \neq \phi$ and $C_2 \neq \phi$.

It is easy to see that any local realization of $M_{new} = C_1 \cup M \cup C_2$ requires at least one boundary wire. If a net in C_1 (resp., C_2) is routed by a boundary wire, no wire corresponding to a net in C_2 (resp., C_1) can be a boundary wire. Therefore, if M_{new} is locally realizable, any such realization has either left or right boundary wires.

In what follows we assume that $|C_1| \geq |C_2|$, that is, $r \geq s$. The other case can

be treated in a similar way. Let $H_1^* = H_1 - \{b_{v_0'}\}$ and $H_2^* = H_2 - \{b_{v_r \oplus 1}\}$ (H_1 and H_2 are defined at the beginning of this subsection). Let $W = \kappa_{v_1'} + \kappa_{v_2'} + \cdots + \kappa_{v_s'}$ (resp., $\kappa_{v_1'} + \kappa_{v_2'} + \cdots + \kappa_{v_s'} - l(M)$) if $type(M) = `x'$ (resp., 'f'). W represents the total number of wires which can pass between all pairs of adjacent terminals in H_1^*. We first consider the cases in which M_{new} is not locally realizable.

Case 1. $type(M) = `f'$, $l(M) = \kappa_{v_s'}$ and $r(M) = \kappa_{q_m}$.

If w_{u_r} (resp., $w_{u_1'}$) does not pass between $b_{v_s'}$ and b_{q_1} (resp., b_{q_m} and b_{v_1}), $w_{u_1'}$ (resp., w_{u_r}) must pass between b_{q_m} and b_{v_1} (resp., $b_{v_s'}$ and b_{q_1}). That is, at least one net in $C_1 \cup C_2$ must be routed by a wire which passes between either $b_{v_s'}$ and b_{q_1} or b_{q_m} and b_{v_1}. Therefore, if $type(M) = `f'$, $l(M) = \kappa_{v_s'}$ and $r(M) = \kappa_{q_m}$, M_{new} is not locally realizable.

Case 2. $r > W + \kappa_{v_0'}$ where v_0' is the integer such that $v_0' \oplus 1 = v_1'$

By Lemma 7, all wires corresponding to the nets in C_1 must pass between terminals in H_1. Since the total number of wires which can pass between $s+1$ pairs of adjacent terminals in H_1 is $W + \kappa_{v_0'}$, M_{new} is not locally realizable.

We now show that $M_{new} = C_1 \cup M \cup C_2$ is locally realizable if M_{new} satisfies the conditions that (i) $l(M) < \kappa_{v_s'}$ or $r(M) < \kappa_{q_m}$ if $type(M) = `f'$, and (ii) $r \leq W + \kappa_{v_0'}$. By Lemma 7, every wire for C_1 passes between two adjacent terminals in H_1. Once such a pair of terminals are determined, the layout of C_2 is automatically determined. For example, if w_{u_r} $\wr \bullet$ $b_{v_s'}$, then $w_{u_1'}$ $\wr \bullet$ b_{v_1}, and if w_{u_x} $\wr \bullet$ $b_{v_{s-y-1}'}$ and $w_{u_{x+1}}$ $\wr \bullet$ $b_{v_{s-y+1}'}$, then $w_{u_{y+1}'}$ $\wr \bullet$ $b_{v_{r-x+1}}$ and $w_{u_{y+2}'}$ $\wr \bullet$ $b_{v_{r-x+1}}$. Thus, it suffices to describe how to route the nets in C_1 in order to show the local realization of M_{new}. We also obtain the parameter values of M_{new}.

First assume that $r > W$. We can route the nets in C_1 in such a way that W wires pass between terminals in H_1^* and the remaining $r - W$ wires are left boundary wires of M_{new}. Note in this case that if $type(M) = 'x'$, we choose an r–realization of M. Since at least one wire passes between every pair of adjacent terminals in H_1^*, at most one wire for C_2 passes between each pair of adjacent terminals in H_2^*. This implies that we can obtain a local realization of M_{new} with $r - W$ left boundary wires, and hence M_{new} is locally realizable. On the other hand, if we realize the nets in M_{new} with right boundary wires, all the wires corresponding to the nets in C_1 must pass between terminals in H_1^*, which is not possible since $r > W$. Therefore, we can not obtain a local realization of M_{new} that has right boundary wires, and hence the parameter values are set as $type(M_{new}) = 'f'$, $l(M_{new}) = r - W$ and $r(M_{new}) = 0$.

Second assume that $r \leq W$. There are three possible cases.

Case 1. $|C_1| > |C_2|$ $(r > s)$.

If $type(M) = 'x'$, we choose an r–realization of M. Since $r \leq W$, we can route the nets in C_1 by wires passing between terminals in H_1^*. Moreover, since $r > s$, those wires can be arranged in such a way that at least one wire passes between every pair of adjacent terminals in H_1^*. For example, see the figures on the left hand side of Fig. 9, where we assume that $\kappa_i = 2$ for all $i = 1, 2, \ldots, n$, and that $|C_1| = 4$ and $|C_2| = 3$. This way of routing results in that at most one wire for a net in C_2 passes between each pair of terminals in H_2^* and $w_{u'_s}$ becomes a right boundary wire. Thus, we can have an r–realization of M_{new} with one right boundary wire. On the other hand, suppose that we route the net $n_{u_1} \in C_1$ as $w_{u_1} \wr \bullet\, b_{v'_1}$. Since $r - 1 \geq s$, the remaining nets in C_1 can still be routed by wires passing between terminals in H_1^* such

that at least one wire passes between every pair of adjacent terminals. This implies that at most one wire for a net in C_2 passes between each pair of adjacent terminals in H_2^*. See the figures on the right hand side of Fig. 9. Thus, we can also have an l–realization of M_{new} with one left boundary wire. Therefore in this case, M_{new} is locally realizable, and the parameter values are set as $type(M_{new}) =$ 'x', $l(M) = 1$ and $r(M) = 1$.

Case 2. $|C_1| = |C_2|$ $(r = s)$ and $type(M) =$ 'x'.

The following way of routing will create an l– or r–realization of M_{new}.

(i) An l–realization: Obtain an l–realization of M and route the nets in C_1 and C_2 as $w_{u_i} \wr \bullet b_{v_i'}$ and $w_{u_{s-i+1}'} \wr \bullet b_{v_{r-i+1}}$ for $i = 1, 2, \ldots, r$ (see Fig. 10 (a)).

(ii) An r–realization: Obtain an r–realization of M and route the nets in C_1 and C_2 as $w_{u_{s-i+1}'} \wr \bullet b_{v_{r-i+1} \oplus 1}$ and $w_{u_i} \wr \bullet b_{v_i' \oplus 1}$ for $i = 1, 2, \ldots, r$ (see Fig. 10 (b)).

Therefore, in this case, the parameter values are set as $Type(M_{new}) =$ 'x', $l(M_{new}) = 1$ and $l(M_{new}) = 1$.

Case 3. $|C_1| = |C_2|$ $(r = s)$ and $type(M) =$ 'f'.

There are at most r pairs of adjacent terminals in H_1^*. Since $r \leq W$, we can route the nets in C_1 by wires passing between terminals in H_1^* such that at least one wire passes between every pair of adjacent terminals. Note that if $l(M) = \kappa_{v_s'}$, then $w_{u_r} \wr \bullet b_{v_s'}$ and $n_{u_1'}$ can be routed as $w_{u_1'} \wr \bullet b_{v_1}$ since $r(M) < \kappa_{q_m}$. This way of routing results in that at most one wire for C_2 passes between each pair of adjacent terminals in H_2^*, and hence M_{new} is locally realizable. For example, see Fig. 11 (a). In this case, we have an r–realization of M_{new} since $w_{u_s'}$ is the right boundary wire. We still need to check the possibility of an l–realization of M_{new}. If $r(M) < \kappa_{q_m}$, we can

obtain an l–realization of M_{new} by routing the nets in C_1 and C_2 as $w_{u_i}\ l \bullet b_{v'_i}$ and $w_{u'_{s-i+1}}\ l \bullet b_{v_{r-i+1}}$ for $i = 1, 2, \ldots, r$. For example, see Fig. 11 (b). If $r(M) = \kappa_{q_m}$, n_{u_r} must be routed as $w_{u_r}\ l \bullet b_{q_1}$ since $w_{u'_1}$ can not pass between b_{q_m} and b_{v_1}. Therefore, exactly s nets from C_1 are required in order that at least one wire passes between every pair of adjacent terminals in H_1^*. However, since at least one wire for a net in C_1 must be a left boundary wire, at most $r-1$ wires are available and hence no wire passes between at least one pair of adjacent terminals in H_1^*. This implies that $r = s \geq 2$ and that there must be at least one pair of adjacent terminals, say b_{v_j} and $b_{v_{j+1}}$, in $H_2^* - \{b_{q_m}\}$ such that $\kappa_{v_j} \geq 2$. Therefore, if $r(M) = \kappa_{q_m}$ and $r = s = 1$ or $\kappa_{v_i} = 1$ for all $i = 1, 2, \ldots, r-1$, we can not have an l–realization of M_{new}. For example, see Fig. 11 (d). Otherwise, we can have an l–realization of M_{new} with one left boundary wire (see Fig. 11 (c)). From the above discussions, M_{new} is locally realizable. If $r(M) = \kappa_{q_m}$ and $r = s = 1$ or $\kappa_{v_i} = 1$ for all $i = 1, 2, \ldots, r-1$, the parameter values are set as $type(M_{new}) =$ 'f', $l(M_{new}) = 0$ and $r(M_{new}) = 1$; otherwise, $type(M_{new}) =$ 'x', $l(M_{new}) = 1$ and $r(M_{new}) = 1$.

In Appendix A–2, we describe a complete procedure for an X–merging operation for the case in which $C_1 \neq \phi$ and $C_2 \neq \phi$, and $|C_1| \geq |C_2|$.

5. Algorithm Description

If $N(\pi)$ itself is a cluster, π is realizable and a CPL–solution is trivially formed. We arbitrarily select a net and route it by a direct wire. The remaining nets can be routed by either a Type DL or Type DR layout. Thus, we assume that $N(\pi)$ consists of three or more maximal clusters.

The algorithm consists of two main phases, the *merging phase* and the *routing phase*. In the merging phase, it tests whether π is realizable or not by using the merging operations described in Section 4. If π is realizable, the algorithm constructs a CPL–solution in the routing phase.

5.1 Merging Phase

We show below an outline of the merging phase.

Step 1. Partition $N(\pi)$ into maximal clusters.

Step 2. Execute the following substeps until a merging operation fails or there remain no merging operands.

(a) Find P– or X–merging operands.

(b) Perform the merging operation.

Step 3. **if** Step 2 results in a single component $M = \{(t_{p_i}, b_{q_j}) \mid 1 \leq i, j \leq n\}$ $(= N(\pi))$ such that $type(M) \neq$ 'f' or $l(M)+r(M) \leq k_{q_n}$, where $[p_1, p_2, \ldots, p_n]$ and $[q_1, q_2, \ldots, q_n]$ are **icseq**s,

then go to the routing phase (π is realizable),

else terminate the algorithm (π is not realizable).

In Step 1, we construct a circular doubly linked list, called CLIST, which initially stores all the maximal clusters in the order of their appearances on the outer circle C_{out}. The contents of CLIST will be changed during the execution of Step 2.

In Step 2, the first merging operands can be found according to Lemmas 5 and 6. Since $N(\pi)$ consists of three or more maximal clusters, if there are no parallel maximal clusters, π is not realizable by Lemma 5, and thus the algorithm terminates. If $N(\pi)$

has two parallel maximal clusters, they are components due to Lemma 6. When a maximal cluster C is found to be a component, its parameter values are determined as follows:

Case 1. If C is a trivial cluster, the only net in C has to be routed by a direct wire and hence we set $type(C) = 'f'$, $l(C) = 0$ and $r(C) = 0$.

Case 2. If C is a nontrivial cluster, any local realization of C needs at least one boundary wire by Lemma 3. Since we can have a local realization of C with exactly one, either left or right, boundary wire as shown in the proof of Lemma 1, we set $type(C) = 'x'$, $l(C) = 1$ and $r(C) = 1$.

Recall the definition of the three parameter values given on page 9. In particular, in Case 2 above, C will be laid out as an l– (resp., r–) realization with $l(C) = 1$ (resp., no) left and no (resp., $r(C) = 1$) right boundary wire. In Step 2, the algorithm selects an arbitrary one among the components thus found as the initial core and proceeds to perform the merging operations.

If merging operands including the current core are found, the algorithm tests whether they can be combined into a larger component which is locally realizable. This is done by executing a merging operation described in Section 4. If the merging operation fails, π is not realizable owing to the argument given in Section 4 and the algorithm terminates. Otherwise, the merging procedure returns a new component M_{new} with new values of parameters $type(M_{new})$, $l(M_{new})$ and $r(M_{new})$. This new component becomes a new core and the merging operands are replaced by M_{new} in CLIST. At this time, the algorithm may find another new component according to the following lemma.

Lemma 8. If there exists a maximal cluster C such that $C \,||\, M_{new}$ or $M_{new} \,||\, C$, C is a component.

Proof. Suppose that π is realizable and let $L_{N(\pi)}$ be any CPL–solution for π. For any subset D of $N(\pi)$, let L_D denote the layout of D in $L_{N(\pi)}$. Assume that L_C does not satisfy the condition of local realizability. Then, using the similar argument as in the proof of Lemma 6, we can show that $N(\pi) = C \cup M_{new}$. Let $M_{new} = \{n_{p_i} = (t_{p_i}, b_{q_j}) \mid 1 \leq i, j \leq m\}$ and $C = \{n_{u_i} = (t_{u_i}, b_{v_{k-i+1}}) \mid 1 \leq i \leq k\}$ such that $[u_1, u_2, \ldots, u_k, p_1, p_2, \ldots, p_m]$ and $[v_1, v_2, \ldots, v_k, q_1, q_2, \ldots, q_m]$ are **icseqs**. Since L_C does not satisfy the condition of local realizability, at least one wire, say w_{u_x}, in L_C passes between two adjacent terminals, say b_{q_j} and $b_{q_{j+1}}$, in $\{b_{q_1}, b_{q_2}, \ldots, b_{q_m}\}$. See Fig. 12. This implies that M_{new} is not the result of an X–merging operation but that of a P–merging operation. Let M' and M'' be the P–merging operands such that $M' \cup M'' = M_{new}$ and $M' \,||\, M''$. Note that $n_{p_1} \in M'$ and $n_{p_m} \in M''$. Let $T_1 = \{t_{u_{x+1}}, t_{u_{x+2}}, \ldots, t_{u_r}, t_{p_1}, t_{p_2}, \ldots, t_{p_{m-j}}\}$, $T_2 = \{t_{p_{m-j+1}}, t_{p_{m-j+2}}, \ldots, t_{p_m}, t_{u_1}, t_{u_2}, \ldots, t_{u_{x-1}}\}$, $B_1 = \{b_{q_{j+1}}, b_{q_{j+2}}, \ldots, b_{q_m}, b_{v_1}, b_{v_2}, \ldots, b_{v_{r-x}}\}$ and $B_2 = \{b_{v_{r-x+2}}, b_{v_{r-x+3}}, \ldots, b_{v_r}, b_{q_1}, b_{q_2}, \ldots, b_{q_j}\}$. Similar to the argument given in the proof of Lemma 4, we can show that every terminal in T_1 (resp., T_2) is connected to a terminal in B_1 (resp., B_2). Since $n_{p_1} \in M'$ and $b_{\pi(p_1)} \in B_1$, all the nets connecting the terminals in B_2 are in M', which contradicts the fact that $n_{p_m} \in M''$. Therefore, L_C satisfies the condition of local realizability. □

If there are no merging operands including the current core for a P– or X–merging operation, the next component in the clockwise direction in CLIST becomes a new core.

Lemma 9. If a component in CLIST becomes the core for the second time, there remain no merging operands.

Proof. If a component is selected as the core twice, all of the other components in CLIST have been the core before at least once. The lemma clearly follows from this fact. □

Merging operations are iteratively performed until a merging operation fails for some merging operands or there remain no merging operands in CLIST. Then, in Step 3, the algorithm tests the realizability of π based on the following theorem.

Theorem 1. π is realizable if and only if all maximal clusters in $N(\pi)$ are merged into a single component $M = \{(t_{p_i}, b_{q_j}) \mid 1 \leq i, j \leq n\}$ $(= N(\pi))$ such that $type(M) =$ 'x' or $l(M) + r(M) \leq \kappa_{q_n}$, where $[p_1, p_2, \ldots, p_n]$ and $[q_1, q_2, \ldots, q_n]$ are **icseq**s.

Proof. Suppose that all maximal clusters are merged into a single component $M = \{(t_{p_i}, b_{q_j}) \mid 1 \leq i, j \leq n\}$ $(= N(\pi))$, where $[p_1, p_2, \ldots, p_n]$ and $[q_1, q_2, \ldots, q_n]$ are **icseq**s. It is easy to see that if $type(M) =$ 'x' or $l(M) + r(M) \leq \kappa_{q_n}$, π is realizable. Assume that the algorithm fails in creating such a component. If $type(M) =$ 'f' and $l(M) + r(M) > \kappa_{q_n}$, π is not realizable since $l(M) + r(M)$ boundary wires of the component have to pass between the same two adjacent terminals b_{q_n} and b_{q_1} and only κ_{q_n} wires can pass between them. Suppose that π is realizable and the algorithm terminates when CLIST has two or more elements. Let $L_{N(\pi)}$ be any CPL–solution for π. At the termination of the algorithm CLIST has at least one maximal cluster whose nets are all routed by indirect wires in $L_{N(\pi)}$; otherwise any two consecutive elements in the list would constitute P–merging operands. Let Y be the set of all such maximal clusters and C be a maximal cluster in Y such that $|M_C|$ is a minimum (M_C

is defined in Section 3). Note that M_C consists of two or more maximal clusters. If M_C itself is an element of CLIST at the termination of the algorithm, the list has X–merging operands and the algorithm would continue, a contradiction. Thus, CLIST has at least two elements which form M_C. Since those elements contain no P–merging operands, one of them is a maximal cluster, say C', in Y. Clearly $|M_{C'}| < |M_C|$, and hence $|M_C|$ would not be a minimum. Therefore, if the algorithm terminates when CLIST has two or more elements, π is not realizable. □

During the merging phase, a directed graph called the *merging tree* is constructed. Initially, the graph has only isolated nodes which correspond to the maximal clusters. If a merging operation succeeds, the algorithm adds to the current graph a new node which corresponds to the resultant component and creates directed edges from this node to the nodes corresponding to the merging operands. Thus, if all the maximal clusters are eventually merged into a single component, the graph becomes a directed tree whose root $M = \{(t_{p_i}, b_{q_j}) \mid 1 \leq i, j \leq n\}$ corresponds to $N(\pi)$, where $[p_1, p_2, \ldots, p_n]$ and $[q_1, q_2, \ldots, q_n]$ are **icseq**s. And, if $type(M) \neq$ 'f' or $l(M)+r(M) \leq \kappa_{q_n}$, the merging tree will be used in the next phase to route the nets in $N(\pi)$. We give an example here. If the algorithm is applied to the instance of Fig. 2 and $\{n_{14}\}$ is first selected as the core of the merging operation, the final merging tree would be as shown in Fig. 13. Let $M = \{n_{p_i} = (t_{p_i}, b_{q_j}) \mid 1 \leq i, j \leq m\}$ be a maximal cluster or a component which was created in the merging phase such that $[p_1, p_2, \ldots, p_m]$ and $[q_1, q_2, \ldots, q_m]$ are **icseq**s. The node corresponding to M in the merging tree is represented by a square as shown in Fig. 13. The numbers at the top (resp., bottom) of the square denote p_1 and p_m (resp., q_1 and q_m) from left to right. If M is a

component, the 3–tuple in the center of the square denotes $(type(M), l(M), r(M))$. If M is a maximal cluster, it is indicated by the symbol 'c' in the center of the square.

Theorem 2. The time complexity of the merging phase is $O(n)$.

Proof. It is easy to find all maximal clusters in $O(n)$ time. Since the number of maximal clusters is at most n, both the construction of CLIST and finding parallel maximal clusters can be done in $O(n)$ time. In Step 2, if the algorithm can not find any merging operands including the current core, it selects a new core. By Lemma 9, all such selections require only $O(n)$ time in total. The total number of merging operations performed is at most $n-1$ because each merging operation reduces the number of elements in CLIST by at least one. Furthermore, one execution of a P–merging (resp., X–merging) operation takes a constant (resp., O(size of the maximal clusters in the merging operands)) time. Since a maximal cluster is involved in a merging operation exactly once, all merging operation can be carried out in $O(n)$ time in total. Therefore, the merging phase can be completed in $O(n)$ time. □

5.2. Routing Phase

Once the merging phase is successfully completed, the nets in $N(\pi)$ are to be routed in the routing phase. The merging tree is now a rooted directed tree that has at most $2n-1$ nodes. Its leaves correspond to the maximal clusters in $N(\pi)$ and its nonleaf nodes correspond to the components that have been found in the merging phase. For convenience, if a node in the tree corresponds to a subset M of $N(\pi)$, we call it node M.

In the routing phase, we first check whether the root $M_r = N(\pi)$ is a component

such that $type(N(\pi)) = `x'$, and if so, we change its parameter values as $type(M_r) = `f'$ and either $l(M_r) = l(M_r)$ and $r(M_r) = 0$ or $l(M_r) = 0$ and $r(M_r) = r(M_r)$. This selection is arbitrary. Then, starting from the root, the algorithm visits every nonleaf node M_f of the merging tree by depth–first search [1]. For each child M of M_f, if $type(M) = `x'$, its parameter values are modified as $type(M) = `f'$ and either $l(M) = l(M)$ and $r(M) = 0$ or $l(M) = 0$ and $r(M) = r(M)$ so as to be consistent with the parameter values of M_f. If M is a maximal cluster, the nets in M are routed. These modifications and routings are made in the following manner.

Case 1. M_f was created by a P–merging operation.

Let M_1 and M_2 be the components such that $M_f = M_1 \cup M_2$ and $M_1 \parallel M_2$. Suppose that $type(M_1) = `x'$. If $l(M_f) = 0$, we change the parameter values of $M_1$ as $type(M_1) = `f'$, $l(M_1) = 0$ and $r(M_1) = r(M_1)$; otherwise we change them as $type(M_1) = `f'$, $l(M_1) = l(M_1)$ and $r(M_1) = 0$. Similarly, if $type(M_2) = `x'$, the parameter values of M_2 are changed as $type(M_2) = `f'$ and either $l(M_2) = l(M_2)$ and $r(M_2) = 0$ or $l(M_2) = 0$ and $r(M_2) = r(M_2)$ depending on whether $r(M_f) = 0$ or not. If M_1 is a maximal cluster, the nets in M_1 are routed in the following way. If $l(M_1) = 0$ and $r(M_1) = 0$, M_1 is a trivial cluster and its only net is routed by a direct wire. Otherwise, the nets in M_1 are routed by either a Type DL or a Type DR layout depending on whether $l(M_1) = 1$ or $r(M_1) = 1$. Similarly, if M_2 is a maximal cluster, the nets in M_2 are routed in the same way as above.

Case 2. M_f was created by an X–merging operation.

Let $M = \{n_{p_i} = (t_{p_i}, b_{q_j}) \mid 1 \leq i, j \leq m\}$, $C_1 = \{n_{u_i} = (t_{u_i}, b_{v_{r-i+1}}) \mid 1 \leq i, j \leq r\}$ and $C_2 = \{n_{u'_i} = (t_{u'_i}, b_{v'_{s-i+1}}) \mid 1 \leq i, j \leq s\}$ be a component and two maximal

clusters, respectively, such that $M_f = C_1 \cup M \cup C_2$ and that $[u_1, u_2, \ldots, u_r, p_1, p_2, \ldots, p_m, u'_1, u'_2, \ldots, u'_s]$ and $[v'_1, v'_2, \ldots, v'_s, q_1, q_2, \ldots, q_m, v_1, v_2, \ldots, v_r]$ are **icseqs**. Without loss of generality we assume that $|C_1| \geq |C_2|$. The case in which $|C_1| < |C_2|$ can similarly be treated. If $C_2 = \phi$, all nets in C_1 are routed by wires passing between b_{q_0} and b_{q_1} where q_0 is the integer such that $q_1 = q_0 \oplus 1$. Note in this case that if $type(M) =$ 'x', the parameter values of M are changed as $type(M) =$ 'f', $l(M) = 0$ and $r(M) = r(M)$. We now assume that $C_2 \neq \phi$. Let $W = \kappa_{v'_1} + \kappa_{v'_2} + \cdots + \kappa_{v'_s}$ (resp., $\kappa_{v'_1} + \kappa_{v'_2} + \cdots + \kappa_{v'_s} - l(M)$) if $type(M) =$ 'x' (resp., 'f'). The following procedure will route the nets in C_1 and C_2 in such a way that the layout is consistent with the parameter values of M_f.

1. **procedure**
2. **begin**
3. **if** $r > W$ **then**
4. **call** *l–realization1* (C_1, M, C_2);
5. **else if** $l(M_f) = 0$ **then**
6. **call** *r–realization* (C_1, M, C_2);
7. **else if** $type(M) =$ 'f' **and** $r(M) = \kappa_{q_m}$ **and** $r = s$ **then**
8. **call** *l–realization2* (C_1, M, C_2);
9. **else**
10. **call** *l–realization3* (C_1, M, C_2);
11. **end**

The subroutines used in the above procedure are described in Appendix A–3. Each

subroutine routes the nets in $C_1 \cup C_2$ so as to produce a layout of M_f which is consistent with its corresponding parameter values. If $type(M) = 'x'$, the parameter values of M are also changed appropriately in each subroutine. For example, the routing phase will construct such a CPL–solution as shown in Fig. 2 for the merging tree of Fig. 13.

The number of nodes in the merging tree is $O(n)$. If a component M_f was created by a P–merging operation, the algorithm, while visiting this node, spends $O(|M_1|)$, $O(|M_2|)$ or $O(|M_1| + |M_2|)$ time, respectively, depending on whether M_1, M_2 or both M_1 and M_2 are maximal clusters. Similarly, if M_f was created by an X–merging operation, the algorithm spends $O(|C_1| + |C_2|)$ time for routing the nets in C_1 and C_2. Therefore, the time complexity of the routing phase is $O(n)$. Since the merging phase takes $O(n)$ time by Theorem 2, we establish our main theorem.

Theorem 3. Given a permutation π of $1, 2, \ldots, n$, a CPL–solution for π can be found, if one exists, in $O(n)$ time. □

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Mass., 1974.

[2] S. Choe, T. Kashiwabara, and T. Fujisawa, *A Permutation Layout with Limited Between-pin Congestion*, Papers of Technical Group on Circuit and Systems of the IECE of Japan, CAS 83–74, 1983.

[3] S. Choe, T. Kashiwabara, and T. Fujisawa, *Restricted Permutation Layout*, Trans. of the IECE of Japan J68E (1985), pp. 269–276.

[4] M. Cutler and Y. Shiloach, *Permutation Layout*, Networks 8 (1978), pp. 253–278.

[5] T. Kashiwabara, K. Itagaki, S. Masuda, and T. Fujisawa, *On Certain Permutation Layout*, Proc. 1985 IEEE Int. Symp. on Circuits and Systems, Kyoto,Japan, 1985, pp. 1043–1046.

[6] R. D. Lou and M. Sarrafzadeh, *General Circular Permutation Layout*, Proc. 26th Allerton Conf. on Communication, Control and Computing, Montecillo, IL, Sept. 1988, pp. 1136–1137.

[7] T. Ozawa, *A Routing Procedure for an IC Module with Many Pins - A Solution to a Circular Permutation Layout Problem*, Networks 15 (1985), pp. 33–48.

[8] C. S. Rim, N. J. Naclerio, S. Masuda, and K. Nakajima, *A Linear Time Algorithm for Circular Permutation Layout*, Proc. 25th Allerton Conf. on Communication, Control and Computing, Montecillo, IL, Sept. 1987, pp. 345–354.

[9] I. Shirakawa, *Some Comments on Permutation Layout*, Networks 10 (1980), pp. 179–182.

[10] S. Tsukiyama and E. S. Kuh, *Double-Row Planar Routing and Permutation Layout*, Networks 12 (1982), pp. 287–316.

Appendix. Procedures for The Merging and Routing Phases

A-1. Procedure for a P-merging Operation

Let M and M' be components defined in Section 4.1 and assume that $M \parallel M'$.

1. **procedure** P-merging(M, M')
2. **begin**
3. $M_{new} \leftarrow M \cup M'$;
4. **if** $type(M) =$ 'f' **and** $type(M') =$ 'f' **and** $r(M) + l(M') > \kappa_{q_m}$ **then**
5. **stop**; /* π is not realizable */
6. **if** $type(M) =$ 'x' **and** $type(M') =$ 'x' **and** $r(M) + l(M') > \kappa_{q_m}$ **then**
7. $type(M_{new}) \leftarrow$ 'x';
8. **else** $type(M_{new}) \leftarrow$ 'f';
9. **if** $type(M) =$ 'x' **and** $r(M) + l(M') \leq \kappa_{q_m}$ **then**
10. $l(M_{new}) \leftarrow 0$;
11. **else**
12. $l(M_{new}) \leftarrow l(M)$;
13. **if** $type(M') =$ 'x' **and** $r(M) + l(M') \leq \kappa_{q_m}$ **then**
14. $r(M_{new}) \leftarrow 0$;
15. **else**
16. $r(M_{new}) \leftarrow r(M')$;
17. **return**(M_{new});
18. **end** P-merging

A-2. Procedure for an X-merging Operation

Let M, C_1 and C_2 be a component and two maximal clusters defined in Section 4.2 which form X-merging operands. Assume that $C_1 \neq \phi$, $C_2 \neq \phi$, and $|C_1| \geq |C_2|$.

1. **procedure** X-merging(C_1, M, C_2)
2. **begin**
3. $M_{new} \leftarrow C_1 \cup M \cup C_2$;
4. **if** $type(M) =$ 'f' **and** $l(M) = \kappa_{v'_s}$ **and** $r(M) = \kappa_{q_m}$ **then**
5. **stop**; /* π is not realizable */
6. $W \leftarrow \kappa_{v'_1} + \kappa_{v'_2} + \cdots + \kappa_{v'_s}$;
7. **if** $type(M) =$ 'f' **then** $W \leftarrow W - l(M)$;
8. **if** $r < W$ **then** /* $|C_1| = r$ */
9. **begin**
10. **if** $r - W > \kappa_{v'_0}$ **then** /* $v'_1 = v'_0 \oplus 1$ */
11. **stop**; /* π is not realizable */
12. $type(M_{new}) \leftarrow$ 'f'; $l(M_{new}) \leftarrow r - W$; $r(M_{new}) \leftarrow 0$;
13. **end**
14. **else if** $type(M) =$ 'f' **and** $r = s$ **and** $r(M) = \kappa_{q_m}$ **and**
($r = s = 1$ **or** $\kappa_{v_x} = 1$ for all $x = 1, 2, \ldots, r-1$) **then**
15. **begin**
16. $type(M_{new}) \leftarrow$ 'f'; $l(M_{new}) \leftarrow 0$; $r(M_{new}) \leftarrow 1$;
17. **end**
18. **else**
19. **begin**
20. $type(M_{new}) \leftarrow$ 'x'; $l(M_{new}) \leftarrow 1$; $r(M_{new}) \leftarrow 1$;
21. **end**
22. **return**(M_{new});
23. **end** X-merging

A-3. Routing for Two Maximal Clusters in an X-merging Operation

Let M, C_1 and C_2 be a component and two maximal clusters, respectively, defined in Section 4.2 such that $M_f = C_1 \cup M \cup C_2$. Assume that $C_1 \neq \phi$, $C_2 \neq \phi$, and $|C_1| \geq |C_2|$. The following four procedures route the nets in $C_1 \cup C_2$ in such a way that the layout of M_f will be consistent with its parameter values. If $type(M) =$ 'x', the parameter values of M are changed as either $type(M) =$ 'f', $l(M) = l(M)$ and $r(M) = 0$ or $type(M) =$ 'f', $l(M) = 0$ and $r(M) = r(M)$ so that the layout of M does not conflict with the wires for $C_1 \cup C_2$. Note that $|C_1| = r$ and $|C_2| = s$ and that each procedure can be done in $O(r + s)$ time.

(a) $r > W$. Note that $r - W$ wires become left boundary wires.

1. **procedure** *l-realization1* (C_1, M, C_2) /* obtains an l-realization of M_f */
2. **begin**
3. **if** $type(M) =$ 'f' **and** $l(M) = \kappa_{v'_s}$ **then**
4. **begin**
5. route $n_{u'_1}$ as $w_{u'_1}$ $l \bullet b_{v_1}$; $h \leftarrow s - 1$;
6. **end**
7. **else** $h \leftarrow s$;
8. $x \leftarrow r$;
9. **if** $h = 0$ **then goto** 19;
10. **for** $i = h$ **to** 1 **step** -1 **do**
11. **begin**
12. **if** $type(M) =$ 'f' **and** $i = s$ **then** $z \leftarrow \kappa_{v'_i} - l(M) - 1$;
13. **else** $z \leftarrow \kappa_{v'_i} - 1$;
14. **for** $j = 0$ **to** z **step** 1 **do**
15. route $n_{u_{x-j}}$ as $w_{u_{x-j}}$ $l \bullet b_{v'_i \oplus 1}$;
16. route $n_{u'_{s-i+1}}$ as $w_{u'_{s-i+1}}$ $l \bullet b_{v_{r-x+z+1} \oplus 1}$;
17. $x \leftarrow x - z - 1$;
18. **end**
19. **for** $i = x$ **to** 1 **step** -1 **do**
20. route n_{u_i} as w_{u_i} $l \bullet b_{v'_1}$;
21. **if** $type(M) =$ 'x' **then**
22. **begin**
23. $type(M) \leftarrow$ 'r'; $l(M) \leftarrow 0$; $r(M) \leftarrow r(M)$;
24. **end**
25. **end** *l-realization1*

(b) $r \leq W$.

```
1.  procedure r-realization (C_1, M, C_2)      /* obtains an r-realization of M_f */
2.  begin
3.    if type(M) = 'f' and l(M) = κ_{v'_s} then begin
4.        route n_{u'_1} as w_{u'_1} ≀ • b_{v_1};  h ← s − 1;
5.      end
6.    else h ← s;
7.    x ← r;  y ← r − h;
8.    for i = h to 1 step −1 do
9.      begin
10.       if type(M) = 'f' and i = s then z ← κ_{v'_i} − l(M) − 1;
11.       else z ← κ_{v'_i} − 1;
12.       if y > z then begin
13.             for j = 0 to z step 1 do
14.               route n_{u_{x−j}} as w_{u_{x−j}} ≀ • b_{v'_i⊕1};
15.             route n_{u'_{s−i+1}} as w_{u'_{s−i+1}} ≀ • b_{v_{r−x+z+1}⊕1};
16.             x ← x − z − 1;  y ← y − z;
17.           end
18.       else begin
19.             for j = 0 to y step 1 do
20.               route n_{u_{x−j}} as w_{u_{x−j}} ≀ • b_{v'_i⊕1};
21.             route n_{u'_{s−i+1}} as w_{u'_{s−i+1}} ≀ • b_{v_{r−x+y+1}⊕1};
22.             x ← x − y − 1;  goto 26;
23.           end
24.     end
25.   go to 31;
26.   if i ≤ 1 then goto 31;
27.   for j = i − 1 to 1 step −1 do begin
28.       route n_{u_x} as w_{u_x} ≀ • b_{v'_j⊕1};  route n_{u'_{s−j+1}} as w_{u'_{s−j+1}} ≀ • b_{v_{r−x+1}⊕1};
29.       x ← x − 1;
30.     end
31.   if type(M) = 'x' then begin
32.       type(M) ← 'r';  l(M) ← 0;  r(M) ← r(M);
33.     end
34. end r-realization
```

(c) $r \leq W$, and $type(M) = 'f'$, $r(M) = \kappa_{q_m}$ and $r = s$. Note that there is at least one integer j, $1 \leq j \leq r-1$, such that $\kappa_{v_j} \geq 2$. Also note that this procedure can be applied only for the case when $r = s \geq 2$.

1. **procedure** *l-realization2* (C_1, M, C_2) /* obtains an l-realization of M_f */
2. **begin**
3. **for** $i = 1$ **to** r **step** 1 **do**
4. **begin**
5. route n_{u_i} as $w_{u_i} \wr \bullet b_{v'_i}$;
6. **if** $\kappa_{v_{r-i}} \geq 2$ **then**
7. **begin**
8. route $n_{u'_{s-i+1}}$ as $w_{u'_{s-i+1}} \wr \bullet b_{v_{r-i+1}}$; route $n_{u'_{s-i}}$ as $w_{u'_{s-i}} \wr \bullet b_{v_{r-i+1}}$;
9. **goto** 14;
10. **end**
11. **else**
12. route $n_{u'_{s-i+1}}$ as $w_{u'_{s-i+1}} \wr \bullet b_{v_{r-i+1}}$;
13. **end**
14. route $n_{u_{i+1}}$ as $w_{u_{i+1}} \wr \bullet b_{v'_{i+1} \oplus 1}$;
15. **if** $i + 1 = r$ **then return**;
16. **for** $j = i + 2$ **to** r **step** 1 **do**
17. **begin**
18. route $n_{u'_{s-j+1}}$ as $w_{u'_{s-j+1}} \wr \bullet b_{v_{r-j+2}}$; route n_{u_j} as $w_{u_j} \wr \bullet b_{v'_j \oplus 1}$;
19. **end**
20. **end** *l-realization2*

(d) $r \leq W$, and $type(M) =$ 'x' or $r(M) < \kappa_{q_m}$ or $r > s$.

```
1.  procedure l-realization3 (C_1, M, C_2)      /* obtains an l-realization of M_f */
2.  begin
3.     h ← s;
4.     if type(M) = 'x' and r = s then h ← s − 1;
5.     if type(M) = 'f' and l(M) = κ_{v'_s} then h ← s − 1;
6.     x ← 2;  y ← r − h − 1;
7.     route n_{u_1} as w_{u_1} l • b_{v'_1};
8.     if y < 0 then
9.        begin
10.          j ← 1;  goto 32;
11.       end
12.    for i = 1 to h step 1 do
13.       begin
14.          route n_{u'_{s−i+1}} as w_{u'_{s−i+1}} l • b_{v_{r−x+2}};
15.          if type(M) = 'f' and i = s then z ← κ_{v'_i} − l(M) − 1;
16.          else z ← κ_{v'_i} − 1;
17.          if y > z then
18.             begin
19.                for j = 0 to z step 1 do
20.                   route n_{u_{x+j}} as w_{u_{x+j}} l • b_{v'_i⊕1};
21.                x ← x + z + 1;  y ← y − z;
22.             end
23.          else
24.             begin
25.                for j = 0 to y step 1 do
26.                   route n_{u_{x+j}} as w_{u_{x+j}} l • b_{v'_i⊕1};
27.                x ← x + y + 1;  goto 31;
28.             end
29.       end
30.    goto 39;
```

31. $j \leftarrow i+1$;
32. **while** $s-j+1>0$ **do**
33. **begin**
34. route $n_{u'_{s-j+1}}$ as $w_{u'_{s-j+1}} \wr \bullet b_{v_{r-x+2}}$;
35. **if** $x>r$ **then goto** 39;
36. route n_{u_x} as $w_{u_x} \wr \bullet b_{v'_j \oplus 1}$;
37. $x \leftarrow x+1$; $j \leftarrow j+1$;
38. **end**
39. **if** $type(M)=$ 'x' **and** $r=s$ **then**
40. **begin**
41. $type(M) \leftarrow$ 'f'; $l(M) \leftarrow l(M)$; $r(M) \leftarrow 0$;
42. **end**
43. **if** $type(M)=$ 'x' **and** $r \neq s$ **then**
44. **begin**
45. $type(M) \leftarrow$ 'f'; $l(M) \leftarrow 0$; $r(M) \leftarrow r(M)$;
46. **end**
47. **end** *l-realization3*

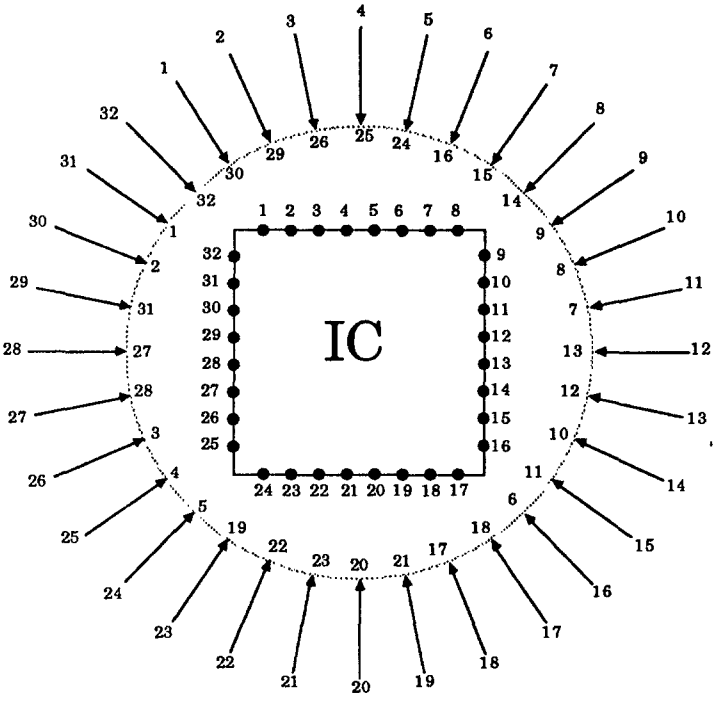


Fig. 1. An application of circular permutation layout in a single layer around an IC (module).

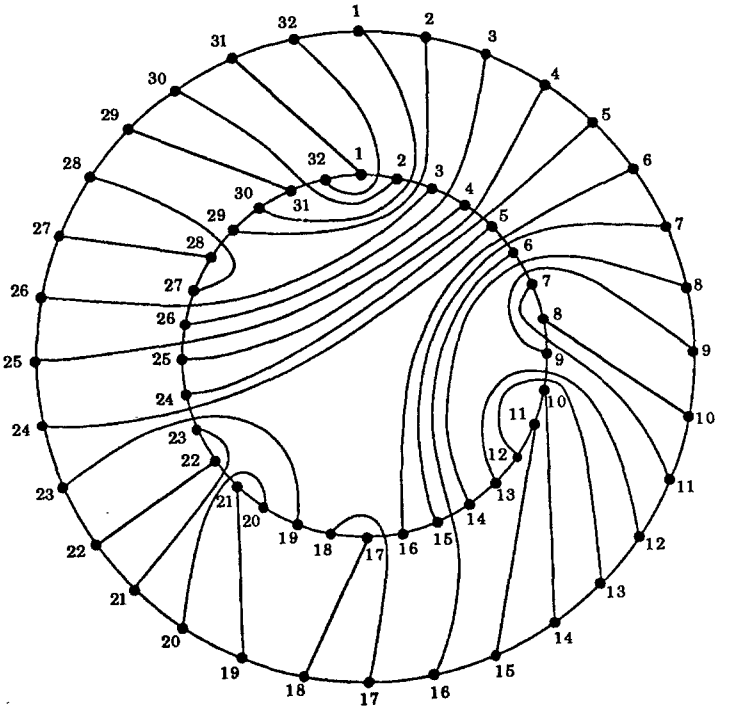
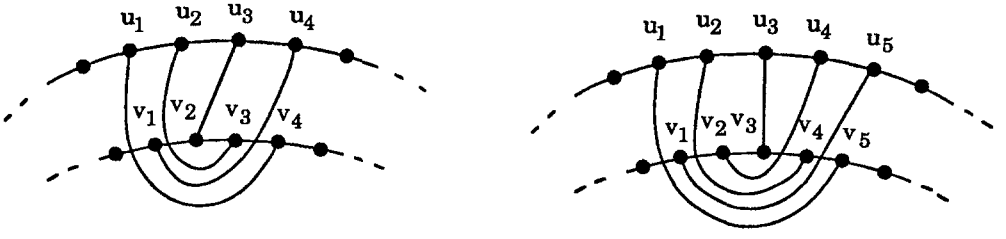
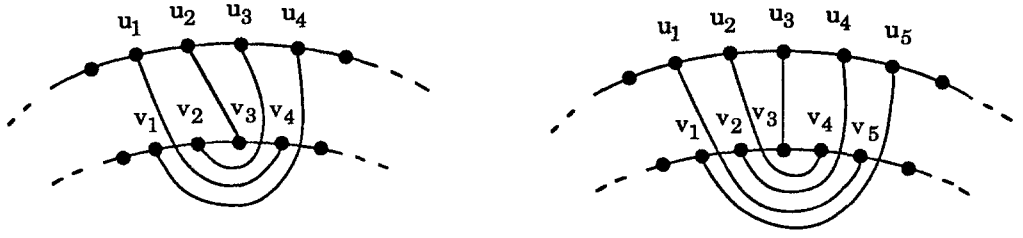


Fig. 2. A CPL-solution for the permutation
π = (30 29 26 25 24 16 15 14 9 8 7 13 12 10 11 6 18 17 21 20 23 22 19 5 4 3 28 27 31 2 1 32).



(a) Type DL layouts.



(b) Type DR layouts.

Fig. 3. Examples of Type DL and Type DR layouts of clusters.

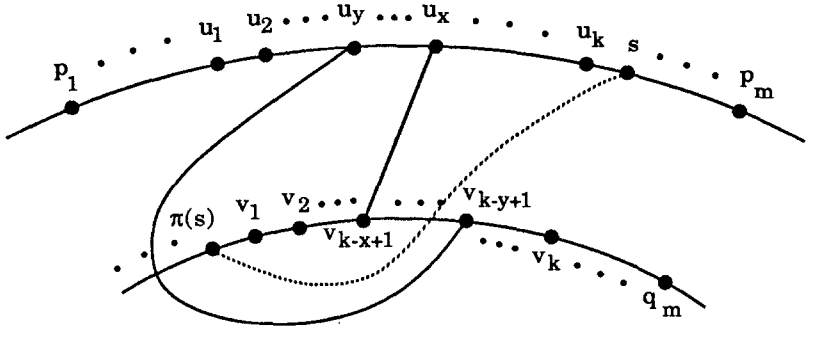


Fig. 4. An illustration for the proof of Lemma 2.

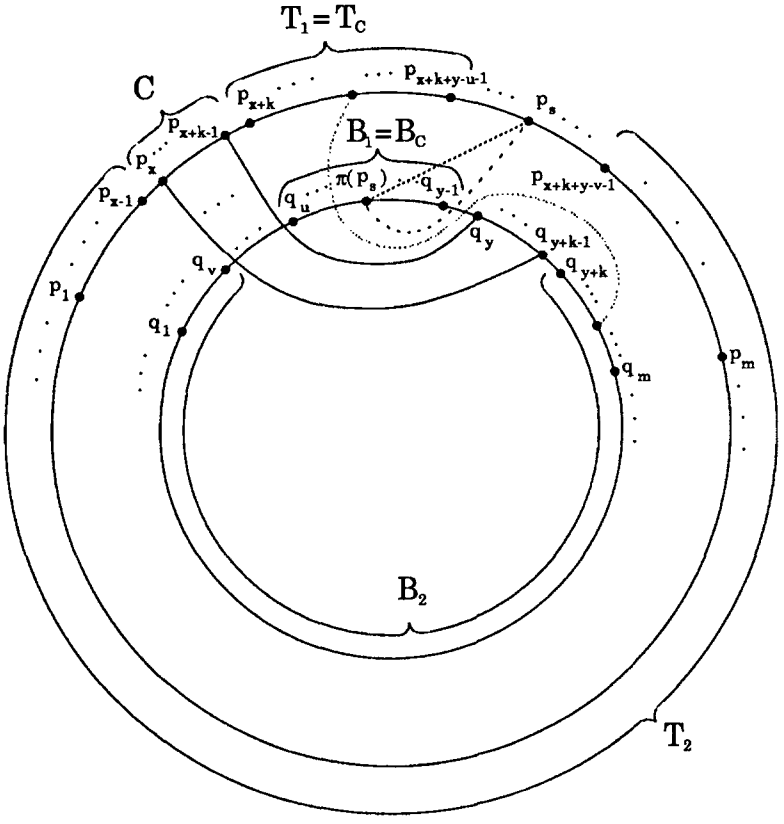


Fig. 5. An illustration for the proof of Lemma 4 with $|T_1| \leq |T_2|$.

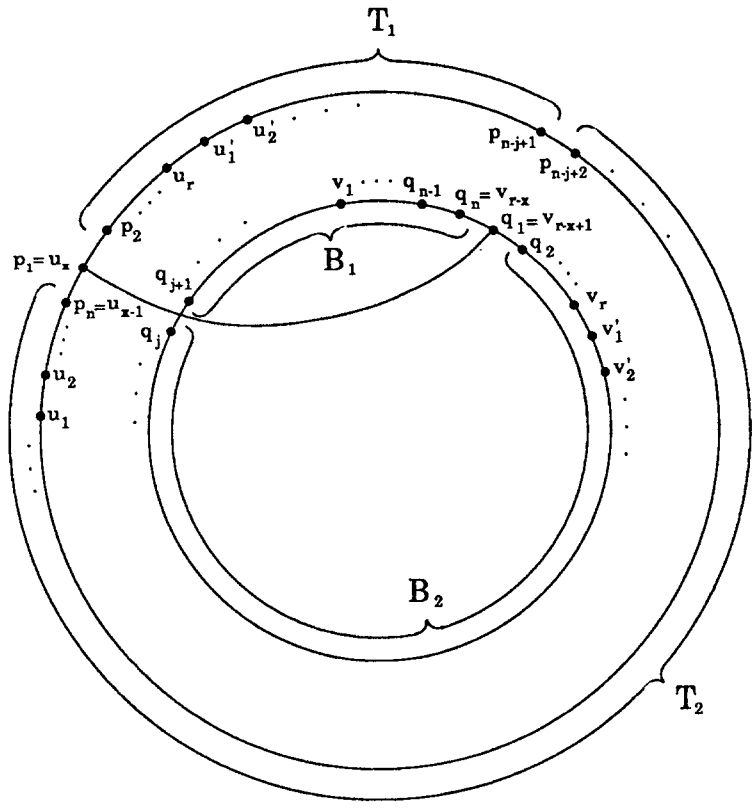
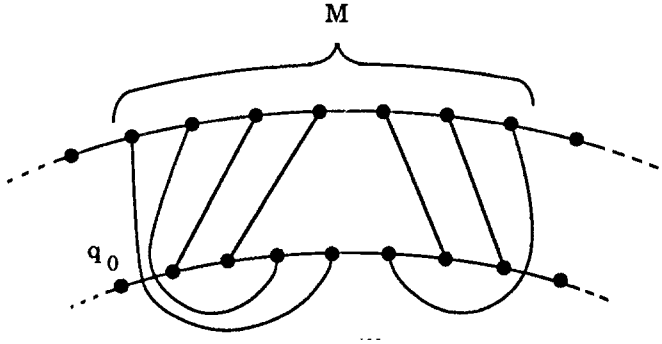
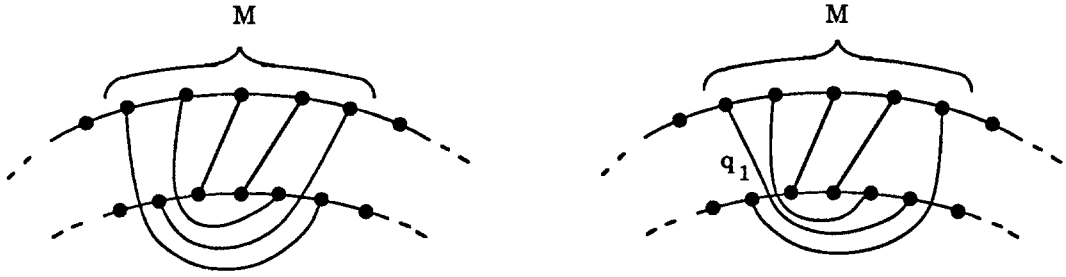


Fig. 6 An illustration for the proof of Lemma 6.



(a) $type(M) =$ 'f', $l(M) = 2$ and $r(M) = 1$ (assume that $\kappa_{q_0} \geq 2$).



(b) $type(M) =$ 'x', $l(M) = 1$ and $r(M) = 1$ (assume that $\kappa_{q_1} \geq 2$).

Fig. 7. Example layouts of components and their parameters.

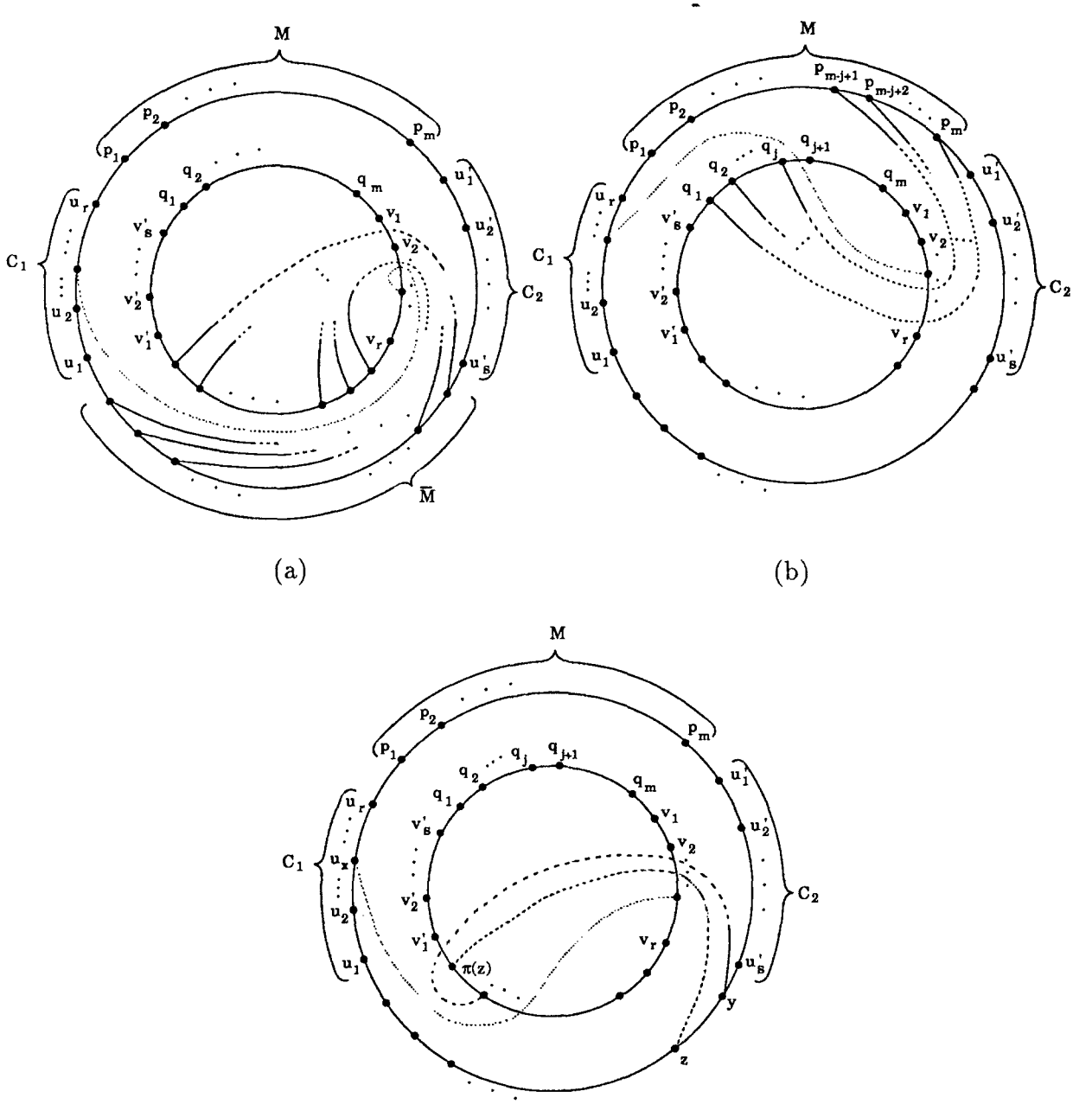
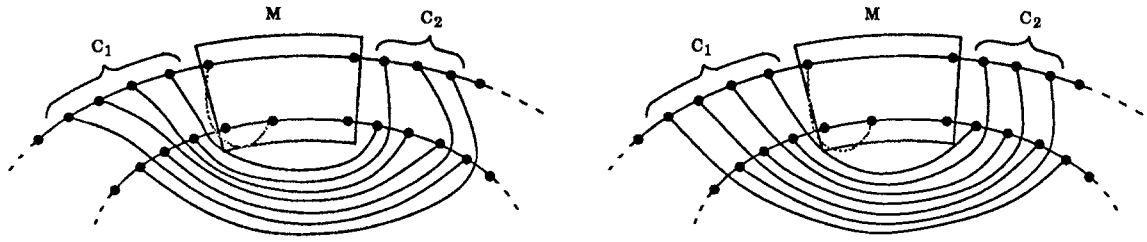
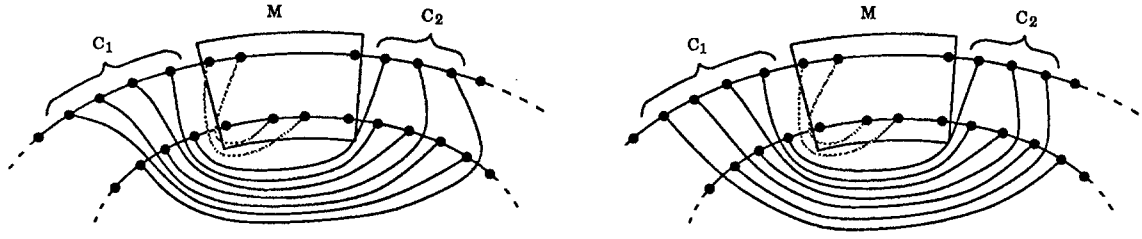


Fig. 8. Illustrations for the proof of Lemma 7.



an r-realization. an l-realization.

(a) $W = 5$, and $type(M) =$ 'f', $l(M) = 1$, and $r(M) \leq 2$.



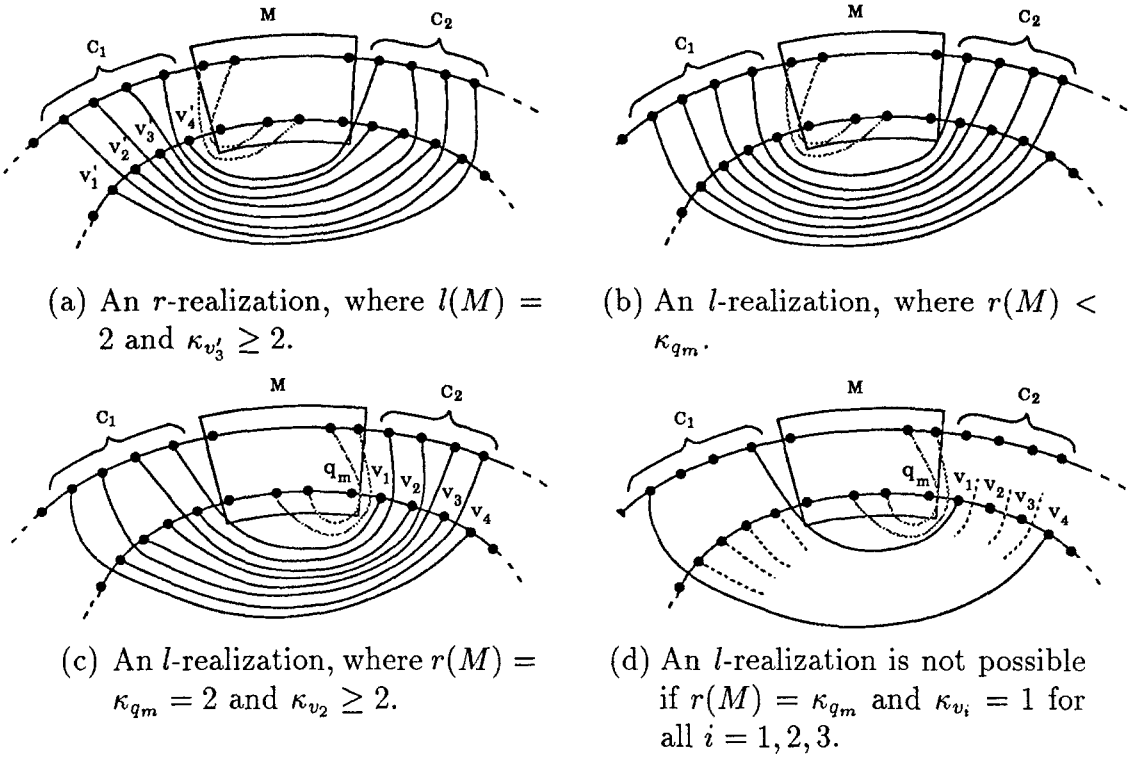
an r-realization. an l-realization.

(b) $W = 4$, and $type(M) =$ 'f', $l(M) = 2$, and $r(M) < 2$.

Fig. 9. Illustrations of an X-merging operation for the case in which $|C_1| \leq W$ and $|C_1| > |C_2|$.

(a) An l-realization of M_{new}. (b) An r-realization of M_{new}.

Fig. 10. Illustrations of an X-merging operation for the case in which $|C_1| \leq W$, $|C_1| = |C_2|$ and $type(M) =$ 'x'.



(a) An r-realization, where $l(M) = 2$ and $\kappa_{v_3'} \geq 2$.

(b) An l-realization, where $r(M) < \kappa_{q_m}$.

(c) An l-realization, where $r(M) = \kappa_{q_m} = 2$ and $\kappa_{v_2} \geq 2$.

(d) An l-realization is not possible if $r(M) = \kappa_{q_m}$ and $\kappa_{v_i} = 1$ for all $i = 1, 2, 3$.

Fig. 11. Illustrations of an X-merging operation for the case in which $|C_1| \leq W$, $|C_1| = |C_2|$ and $type(M) = 'f'$.

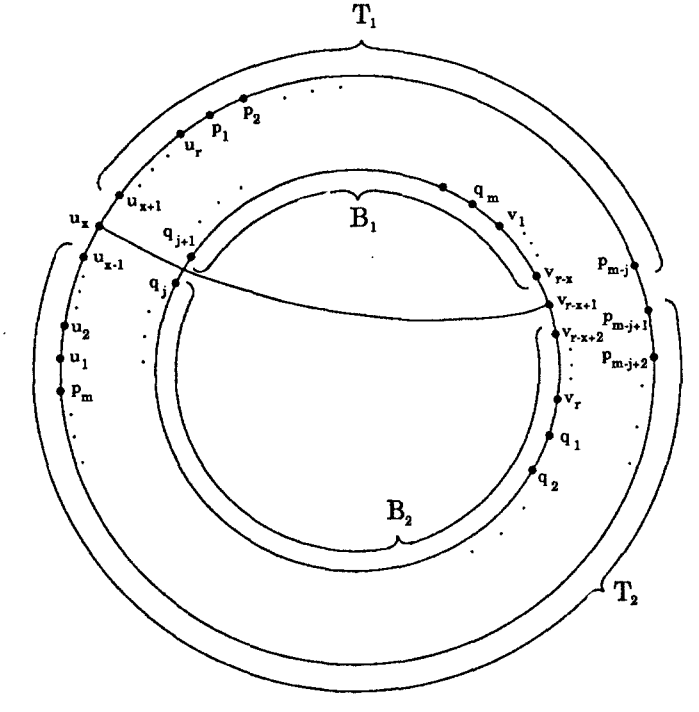


Fig. 12. An illustration for the proof of Lemma 8.

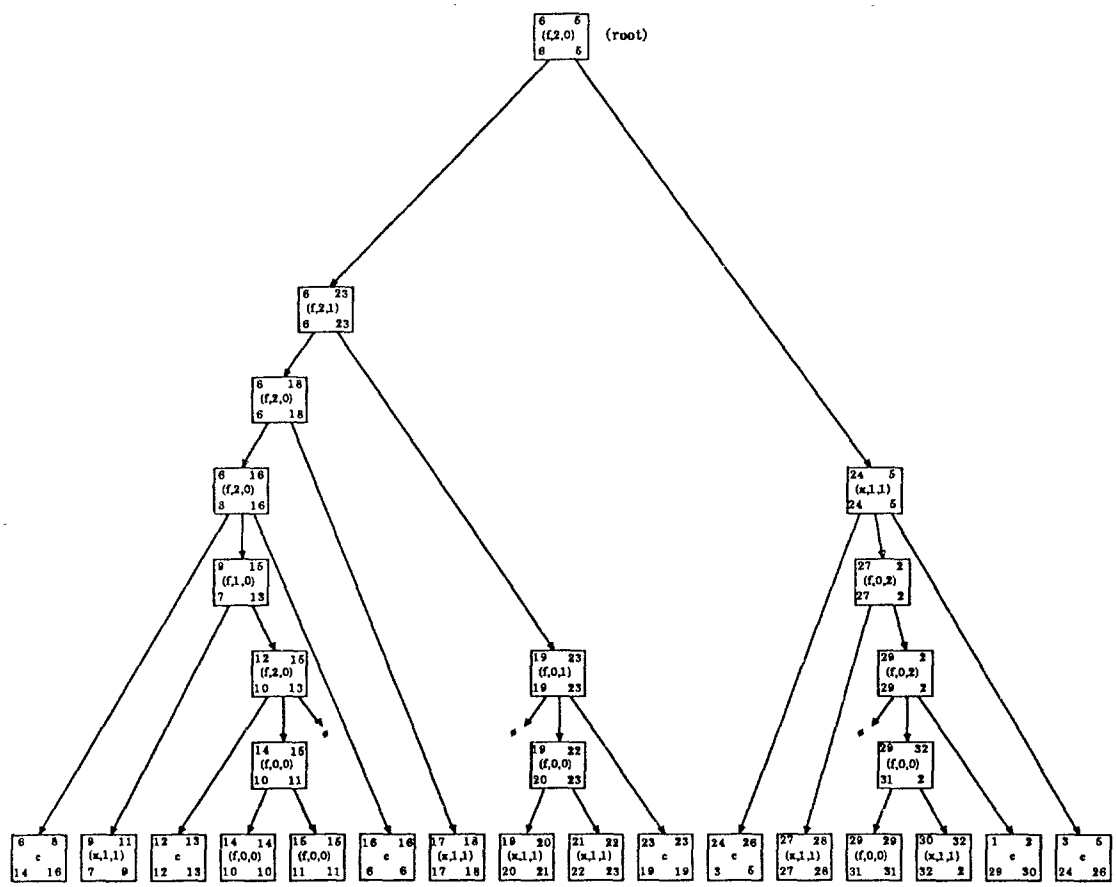


Fig. 13. A result of the merging phase for the instance of Fig. 2.