PLEASURE: A COMPUTER PROGRAM FOR SIMPLE/MULTIPLE CONSTRAINED/UNCONSTRAINED FOLDING OF PROGRAMMABLE LOGIC ARRAYS

by

G. De Micheli and A. Sangiovanni-Vincentelli

Memorandum No. UCB/ERL M82/57

9 August 1982

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

PLEASURE : A Computer Program for Simple/Multiple Constrained/Unconstrained Folding of Programmable Logic Arrays [1]

Giovanni De Micheli

Alberto Sangiovanni-Vincentelli

Department of EECS-University of California at Berkeley

ABSTRACT

Programmable logic arrays are important building blocks of VLSI circuits and systems. We address the problem of optimizing the silicon area and the performances of large logic arrays. In particular we describe a general method for compacting a logic array defined as **multiple row and column folding** and we address the problem of interconnecting a PLA to the outside circuitry. We define a **constrained optimization problem** to achieve minimal silicon area occupation with constrained positions of electrical inputs and outputs. We present a **new computer program, PLEASURE,** which implements several algorithms for multiple and/or constrained PLA folding.

[1] This research has been sponsored by Harris and IBM corporations and NSF under subcontract # 392741C-1.

1. INTRODUCTION

Very Large Scale Integrated Circuits and Systems are so complex that structured design techniques are often used to ensure electrical correctness while maintaining a reasonable design time. Array logic has been used extensively in VLSI design and Programmable Logic Arrays have proved to be an effective means to implement multiple output switching functions [1] [2].

The PLA implementation of a switching function can be partitioned into three tasks : **functional design, topological design, and physical design.** Functional design consists of translating a set of Boolean equations into a set of two-level sum-of-products logical implicants. In general, this step is followed by a logic minimization , in order to reduce the number of implicants and literals. Logic minimizers are effective tools for this task [3][4]. Topological design involves the transformation of the set of implicants into a topological representation of the PLA structure, such as a symbolic table or a stick diagram. The physical design is the translation of the topological representation into the mask layout according to an implementation technology.

In this paper we address the problem of optimizing the area used by a PLA, by means of row and column folding [5]. Wood presented for the first time a folded PLA implementation in [6], and Hachtel et al. an algorithm for PLA folding in [7]. The technique reported in [6] and [7] is referred here to as **simple folding.** Simple folding aims at determining a permutation of the rows (and/or columns) of the array which permits a maximal set of column pairs (and/or row pairs) to be implemented in the same column (row) of the physical array. Folding comes in two flavors : **column folding** and **row folding** . Since large arrays are usually very sparse , a considerable area reduction can be achieved by folding rows and columns.

A generalization of simple folding is **multiple folding** . The objective of multiple column (and/or row) folding is to determine a permutation of the rows (and/or columns) of the PLA which allows to implement in each column (and/or row) of the physical array a set of logic columns (rows). From the description given above , it is clear that multiple folding contains simple folding as a special case. Thus, the area saving achieved by this technique can always be made better than (or, in the worst case, equal to) the one achieved by simple folding. Note that if simple folding is used , the area of the PLA can be reduced at most to 25% , no matter what the sparsity of the personality of the PLA is. If multiple folding is used, we are limited only by the sparsity structure of the PLA.

Greer proposed for the first time a multiple row folded PLA implementation in [8] and called it segmented array. Paillotin and Chuquillanqui et al. presented multiple column folded arrays in [9] and in [10]. A taxonomy of the folding techniques for PLA is reported in [11].

All existing folding techniques have a major drawback. The connection of a folded PLA to the outside circuitry may involve complex and area-consuming routing , because the positions of the inputs and the outputs of a folded array are permuted by the folding algorithm. In order to use effectively PLA folding for VLSI design , it is crucial to allow the positions of inputs and outputs to be constrained.

In this paper we present : i) **a new algorithm for constrained multiple folding** , that allows to compact PLA area while ensuring easy routing of the folded array ; ii) two PLA **architectures** to implement effectively multiply-folded PLAs ; iii) **a general folding computer program , PLEASURE** , which implements the new folding algorithms to accomplish simple and multiple , constrained and unconstrained row and column folding.

2. MULTIPLE FOLDED PLA IMPLEMENTATION

An unfolded PLA has the general structure shown in Fig. 2.1, and can be implemented both in bipolar and MOS technology. We refer in this paper to the NOR-NOR nMOS implementation presented in [12] as the standard PLA architecture.

The implementation of simple column (and/or) folded PLA is straightforward. since at most two columns (rows) are folded together and connection to the outside circuitry can be done from the top or the bottom of the array. (Fig 2.2) [5] [6]. The implementation of a multiply folded PLA is more complex. We deal first with the implementation of multiply column-folded logic arrays.

The implementation of several logic columns in the same physical location requires the physical (metal,poly or diffusion) columns be split into segments (fig 2.3). Therefore a path must be provided to route input and output signals to/from the split physical columns inside the array. Thus standard PLA architectures cannot be used to implement multiply column-folded PLAs. Several authors [8] [10] [13] have proposed different architectures for multiply folded arrays. We consider the following two structures , which can be implemented in nMOS or cMOS technology.

The first architecture is shown in fig. 2.4. It requires two levels of metal (polysilicon), in addition to the usual levels of poly (metal) and diffusion. The PLA is implemented using two arrays (the AND plane and the OR plane) personalized by MOS transistors. Input signals run vertically in the input columns of the AND plane , product terms run horizontally in the rows of both planes and output columns run vertically in the OR plane. Two levels of interconnect are used for these rows and columns , in addition to ground diffusion rows and columns. The third level of interconnect (second metal or second poly level) is used to run horizontal **connection-rows** above the product term rows to route the input and output signals to/from the input and output columns segments to the outside circuitry.

An alternative architecture supports multiple folding with only one level of metal , poly and diffusion. Input and output signals are routed inside/outside the array by connection-rows parallel and alternated to the product term rows and implemented on the same level. This structure is simpler than the previous one but the area used by a multiply folded PLA is larger (fig. 2.5).

It is important to note that PLAs implemented with either structure are essentially circuit blocks through which input and output busses run straight in the connection-rows. They are therefore excellent building blocks of a regular and structured VLSI design methodology.

Moreover it is important to point out that column folding induces a permutation of product terms and connection-rows. While product term rows provide connection internal to the PLA only , connection-rows join the array to the outside circuitry and their ordering is essential to an optimal routing of the PLA to the other functional blocks of the circuit.

We therefore define a multiple constrained column folding problem. The goal of multiple constrained folding is to compact the PLA area subject to an ordering of the connection-rows. Constrained multiple folding is necessary , for example , for an area-effective compaction of PLAs implementing switching functions whose inputs and outputs are signal data busses inside a VLSI processor.

We address two constrained column folding problems : **column folding with ordered connection-row assignment** and **column folding with bounded connection-row assignment.** In the former problem, each PLA input (and/or output) column is given a position index. Folding is constrained so that connection-rows can be positioned according to the sequence of indexes of the connected columns. as shown in fig. 2.6. In the latter, each input (and/or output) is given an upper and a lower bound on the position of the contacted connection-row. Folding is constrained so that each connection-row can be assigned to a position with an index satisfying the given bounds (fig.2.7).

Unconstrained multiple row folded PLAs can be implemented with a single-poly , single-metal technology [12]. Row folding induces a permutation of input and output columns, which leads to a segmented array , consisting of a sequence of AND and OR planes. This may be a technological drawback,because product terms require area-consuming connections between adjacent planes , in addition to an increased complexity of input and output routing.

Simple row folding may be constrained so that the folded array shows an AND-OR-AND or an OR-AND-OR structure [11]. In this case input or output signals can be routed to both external planes by connection-rows.

On the other hand multiple row folding leads to a segmentation of the array into more than three planes [8] [14]. Since routing of the columns of the internal planes may be difficult , we introduce a new multiple constrained row folding problem : **row folding with bounded column assignment** . Each column is given a left and right bound and row folding is constrained so that each column can be assigned to a position within the bounds.

Multiply row and column-folded arrays can be implemented with the described architectures , provided that only columns in the external planes are multiply folded. To connect a multiple row and column folded array effectively,it is important to be able to determine which signals are routed to the external planes through connection-rows and which are routed from the top and the bottom of the array.

The related constrained multiple row and column folding problem consists of constraining the fold so that input and output signal can be routed from the desired (left,right, top,bottom) direction.

3. GRAPH THEORETIC INTERPRETATION OF THE MULTIPLE FOLDING PROBLEM

We concentrate our attention on a topological representation of a PLA. The following definitions are a generalization of those given in [7]. A logic array is described by a personality matrix. For the sake of generality, we assume that the $(i, j)^{th}$ entry of the personality matrix is zero if the $(i, j)^{th}$ location of the physical array is occupied by interconnect only. Fig. 3.1 shows the personality of the PLA sketched in Fig. 2.1. Let $\{c_i, \; i = 1, 2, \cdots, nc\}$ ($\{r_i, \; i = 1, 2, \cdots, nc\}$) be the set of columns (rows) of the personality matrix. Each column is labeled **input (output),** if it carries an input (output) signal in the physical array. A maximal set of adjacent input (output) columns is called **input array** or **AND plane (output array** or **OR plane)**. Let $R(c_i)(C(r_i))$ be the set of rows (columns) with a nonzero entry in the i^{th} column (row) of the personality matrix. Two columns c_i, c_j (rows r_i, r_j) are **disjoint** if $R(c_i) \cap R(c_j) = \phi$ $(C(r_i) \cap C(r_j) = \phi)$. A **column-folding list (row-folding list)** is a set of either input or output disjoint columns $f_i = \{c_{i,1}, c_{i,2}, \cdots c_{i,n}\}$ (rows $f_i = \{r_{i,1}, r_{i,2}, \cdots r_{i,n}\}$). An **ordered column-folding list** $o_i = (c_{i,1}, c_{i,2}, \cdots c_{i,n})$ (**ordered row-folding list** $o_i = (r_{i,1}, r_{i,2}, \cdots r_{i,n})$) is a column (row) folding list whose elements are ordered. A **column/row-folding set** is a set of disjoint column/row-folding lists $F = \{f_1, f_2, \cdots, f_k\}$ and **ordered column/row-folding set** is a set of disjoint column/row ordered folding lists $O = \{o_1, o_2, \cdots, o_k\}$. Let U be the set of unfolded columns (rows), i.e. $U = \{c \mid \nexists k \text{ s.t. } c \in o_k\}$ $(U = \{r \mid \nexists k \text{ s.t. } r \in o_k\})$. The column (row) cardinality of a folded PLA is $C(O) = |O| + |U|$ $(R(O) = |O| + |U|)$. An ordered folding list of columns (rows) induces a set $QR(O)$ $(QC(O))$ of ordering relations among the rows (columns):

$$QR(O) = \{r_x < r_y \mid r_x \in R(c_{i,j}) \; ; \; r_y \in R(c_{i,j+1}) \; ; c_{i,j}, c_{i,j+1} \in o_i ; o_i \in O\}$$

$(QC(O) = \{c_x < c_y \mid c_x \in C(r_{i,j})\ ;\ c_y \in C(r_{i,j+1})\ ; r_{i,j}, r_{i,j+1} \in o_i; o_i \in O\})$

Let $QR^+(O)$ $(QC^+(O))$ be the transitive closure of $QR(O)$ $(QC(O))$ [15]. A column (row) ordered folding set is **implementable** if $QR^+(O)(QC^+(O))$ is a partial order of the set Z^+.

The optimal unconstrained column (row) folding problem can be stated as follows:

Find an implementable ordered folding set that minimizes the column (row) cardinality of the PLA.

Remark 3.1 : In the simple folding case $|U|$ = (initial column/row set cardinality) $-2|O|$. Hence the optimal unconstrained simple folding problem is to find an implementable ordered folding set with maximum cardinality. ■

We introduce a graph theoretic interpretation of the multiple folding problem in order to gain a better insight into the problem and to study heuristics for the related algorithm. We consider column folding first. According to [7], we define **column-intersection graph** $G(V, E)$ a graph whose nodes $v \in V$ are in one-to-one correspondence with the columns of the logic array and the set E is defined as $E = \{v_i, v_j \mid R(c_i) \cap R(c_j) \neq \phi\}$. Given an ordered column-folding set O, we introduce an associated mixed graph $G(O) = (V, E, A(O))$. A mixed graph $G(V, E, A)$ is a graph with two sets of edges, a set of undirected edges E and a set of directed edges A. V and E are defined as in the column-intersection graph. $A(O)$ is defined as:

$$A(O) = \{v_{i,k}, v_{i,k+1} \mid (c_{i,1}, c_{i,2}, \cdots c_{i,k}, c_{i,k+1}, \cdots c_{i,n}) \in O; k = 1, 2, \cdots n-1\}$$

We define *χ-path* in $G(V, E, A(O))$, a directed path $\chi = [v_1, v_2, \ldots .v_p]$ such that:

i) the first edge in χ is directed and the last undirected; i.e. $(v_1, v_2) \in A(O)$ and $\{v_{p-1}, v_p\} \in E$

ii) every undirected edge in χ is followed by a directed edge; i.e. $\{v_i, v_{i+1}\} \in E \rightarrow (v_{i+1}, v_{i+2}) \in A(O) \ \ \forall i = 1, 2, \cdots, p-2$

Example 3.1 : For the PLA sketched in fig. 2.1 and the ordered folding set $O = \{o_1\}$; $o_1 = (c_{10}, c_7, c_9)$, the associated mixed graph is shown in fig 3.2 and the partially folded array in fig.3.3. A χ*–path* is $[v_{10}, v_7, v_9, v_1]$. ■

We define " χ*–cycle* in $G(V, E, A(O))$ a closed χ*–path* having at least two undirected edges.

Theorem 3.1: An ordered column-folding set O is implementable if and only if the induced mixed graph $G(V, E, A(O))$ has no χ*–cycles*.

The proof is reported in the appendix.

Remark 3.1: Theorem 3.1 allows to verify the existence of a row ordering compatible with a column ordered folding set by checking relations among columns only. This procedure is much simpler (and therefore much faster to be executed on a digital computer) than to verify directly cyclic relations in $QR^+(O)$. ■

Remark 3.2: The graph interpretation and Theorem 3.1 applies "mutatis mutandis" to the multiple unconstrained row folding problem. In this case $G(V, E)$ is the row-intersection graph and $G(V, E, A(O))$ is the mixed graph obtained by adding to $G(V, E)$ the set of directed edges:

$$A(O) = \{v_{i,k}, v_{i,k+1} | (r_{i,1}, r_{i,2}, \ldots, r_{i,k}, r_{i,k+1}, \ldots, r_n) \in O; k = 1, 2, \ldots, n-1)$$

■

A graph interpretation of unconstrained row and column folding is more complex, because it involves bookkeeping of the ordering relations among rows and among columns. For this problem the information contained in the column and row intersection graphs is not sufficient.

Example 3.2 : Consider the partially column folded array shown in fig. 3.3. We question the implementability of the array after folding row r_5 with column r_6. The folded array is clearly not implementable, even though it does not introduce any cycle in both intersection graphs. ■

We introduce therefore the row constraint graph G_R and the column constraint graph G_C which are the directed graphs corresponding to the transitive closure relations $QR^+(O_C)$ and $QR^+(O_R)$ induced by the column and row folding sets O_C and O_R [11]. By definition, the ordered folding sets O_R and O_C are implementable if graphs G_R and G_C are acyclic.

4. AN ALGORITHM FOR MULTIPLE PLA FOLDING

The optimal multiple PLA folding problem was shown to be NP-complete in [16]. We therefore propose a heuristic algorithm that can be considered an extension of the simple folding algorithm presented in [5].

We consider first the multiple column folding problem. The ordered column folding set and the mixed graph $G(V, E, A(O))$ are constructed by the algorithm. At each step the algorithm tries to increase the cardinality of the folded column set and verifies the implementability of the folding by checking that the mixed graph has no χ*-cycle*.

A conceptual description of the algorithm is the following:

MASTER ALGORITHM

Step 0: Initialize the folding procedure

Step 1: If the set of columns which have not been processed is empty, stop. Else select a pair of unfolded disjoint columns or an unfolded column and a column folding list as folding candidates.

Step 2: If the fold induces χ*-cycle* in graph $G(V, E, A(O))$, reject it and go to Step 1.

Step 3: If folding has secondary constraints and constraints are not satisfied, reject the fold and goto Step 1.
(This step is performed by the algorithms described in Section 5.)

Step 4: Fold the candidates, modify the PLA accordingly. Go to Step 1.

■

A detailed description of the algorithm for simple column folding is given in [5]. In this section we will concentrate on the generalization to multiple folding.

and on the the procedure for multiple folding candidate selection.

The selection of the candidate columns for multiple folding can be done according to one of the following folding patterns:

1) a new folding list can be formed by folding two unfolded columns.

2) an unfolded column can be folded on top (bottom) of an existing folding list.

3) a folding list can be "opened" and an unfolded column can be folded "by insertion" into an existing folding list.

A selection of the folding pattern and candidate column is done at each step according to a heuristic strategy.

Let us define first the set of descendants $D(v)$ (ancestors $A(v)$) of a vertex V as follows:

a vertex d is **descendant** of v if there is a χ*-path* from v to d.

a vertex a is **ancestor** of v, if v is descendant of a.

We define a adjacency set $ADJ(v)$ of a vertex v, the set of vertices connected to v by an undirected edge. By definition, we consider every vertex adjacent to itself.

We define pseudo-descendants $\tilde{D}(v)$ of a vertex v the union of the adjacency set of v and the descendant sets of each vertex adjacent to v.

$$\tilde{D}(v) = \bigcup_{\tilde{v} \in ADJ(v)} D(\tilde{v}) \cup ADJ(v)$$

Remark 4.1 : It follows from Theorem 3.1 that for each pair of consecutive columns in an implementable ordered folding list, the corresponding vertices v_1 and v_2 are such that:

$$ADJ(v_2) \cap A(v_1) = \phi$$

■

Let us consider now the selection strategy for folding pattern 1.

Example 4.1 : When two columns, say c_1 and c_2, are folded, a directed edge (v_1, v_2) is added to $A(O)$. Hence a *χ-path* joins v_1 to each vertex in $\tilde{D}(v_2)$. Therefore all pseudo-descendants $\tilde{D}(v_2)$ of v_2 are descendants of v_1.

$$D(v_1) \leftarrow D(v_1) \cup \tilde{D}(v_2)$$

Moreover, since a *χ-path* joins each ancestor of v_1 to v_1, the descendants of v_1 are descendants of each ancestor of v_1

$$D(\tilde{v}) \leftarrow D(\tilde{v}) \cup D(v_1) \quad \forall \tilde{v} \in A(v_1)$$

It follows that an upper bound on the number of ancestor-descendant relations induced by the column folding is :

$$\rho_1 = |A(v_1)|\,|\tilde{D}(v_2)|$$

■

It is reasonable to conjecture that the fewer relations are induced, the lower is the probability of finding *χ-cycles* at further steps of the algorithm. Hence a good choice for a candidate folding pair v_1, v_2 is the one for which ρ_1 is minimal. Unfortunately $\frac{n(n-1)}{2}$ candidate pairs have to be tried to find the minimum ρ_1 for an array with n unfolded columns. This procedure is too time consuming for large arrays. Therefore, an alternative selection strategy is used: select the candidate folding pair (v_1, v_2) such that:

$$v_1 = arg \min_{v \in \bar{V}} |A(v)|$$

$$v_2 = arg \min_{v \in \bar{V}} |\tilde{D}(v)|$$

where $\bar{V} \subseteq V$ is the vertex subset corresponding to the unfolded columns.

Similar considerations apply to the candidate selection according to folding pattern 2. When a column c_1 is folded on top of an ordered folding list $(c_{2,1}, \cdots, c_{2,n})$, a directed edge $(v_1, v_{2,1})$ is added to $A(O)$. Hence a *χ-path* joins v_1 to each vertex v_k, such that $v_k \in \tilde{D}(v_{2,1})$. Therefore an upper bound on the number of ancestor-descendant relations induced by the column fold is:

$$\rho_2 = |A(v_1)|\,|\widetilde{D}(v_{2,1})|.$$

Conversely when a column c_2 is folded on the bottom of an ordered folding list $(c_{1,1}, c_{1,2}, \cdots, c_{1,n})$ an oriented edge $(v_{1,n}, v_2)$ is added to $A(O)$. Hence a *χ-path* joins every vertex $A(v_{1,n})$ to every vertex in $\widetilde{D}(v_2)$. Therefore, an upper bound on the number of ancestor-descendant relations induced by the column fold is:

$$\rho_2 = |A(v_{1,n})|\,|\widetilde{D}(v_2)|$$

The strategy for candidate selection according to folding pattern 2 is based on the same considerations used for folding pattern 1.

A slightly different strategy is used for candidate selection according to folding pattern 3.

Example 4.2 : Consider the PLA shown in fig. 2.1. Let us suppose that column c_7 is folded into the folding list $o_1 = (c_{10}, c_9)$ to give (c_{10}, c_7, c_9), as shown by fig. 3.3. The ancestors of c_7 become ancestors of c_9 and the ancestors of c_{10} become ancestors of c_7. ■

In the general case suppose that column $\bar{c}$ is folded into a folding list $(c_{i,1}, c_{i,2}, \cdots, c_{i,n})$ to give $(c_{i,1}, c_{i,2}, \cdots, c_{i,k-1}, \bar{c}, c_{i,k}, \cdots, c_{i,n})$. An oriented edge joins vertex $v_{i,k-1}$ to $\bar{v}$ and $\bar{v}$ to $v_{i,k}$. Hence the ancestors $A(\bar{v})$ become ancestors of the vertices in $\widetilde{D}(v_{i,k})$ and the ancestors $A(v_{i,k-1})$ become ancestors of the vertices in $\widetilde{D}(\bar{v})$. Therefore, an upper bound on the number of ancestor-descendant relationsis:

$$\rho_3 = |A(v_{i,k-1})|\,|\widetilde{D}(\bar{v})| + |A(\bar{v})|\,|\widetilde{D}(v_{i,k})|$$

Unfortunately the computation of the minimum ρ_3 may be too time consuming for large arrays. Hence we find first the candidate for insertion as:

$$\hat{v} = \underset{v \in \overline{V}}{arg\ \min} (\ |\widetilde{D}(v)| + |A(v)|\ )$$

and then the folding list and the insertion position such that :

$$\hat{\rho}_3 = |A(v_{i,k-1})|\,|\tilde{D}(\hat{v})| + |A(\hat{v})|\,|\tilde{D}(v_{i,k})|$$

is minimal.

When the "best" folding candidates have been selected according to the three folding patterns, the selection of the folding pattern is based on a weighted comparison of the upper bounds ρ_i , i = 1, 2, 3. Weighting factors allow to privilege a folding pattern with regard to the others, as, for example, multiple folding versus simple folding.

Remark 4.2: The Master Algorithm and the candidate selection strategy applies "mutatis mutandis" to the multiple unconstrained row folding problem. ■

The Master Algorithm is used for multiple row and column folding also. Order relations induced by the folds are described by the row constraint and column constraint graphs. A candidate fold is rejected at Step 2 of the algorithm if it induces a direct cycle in any of the two graphs. The folding candidate selection strategy is similar to the one used for column folding, provided that some definitions are changed to be compatible with the different graph representation.

For this problem, a vertex d is descendant of v if there is a direct path from v to d; the adjacency set of a vertex is not defined and the pseudo-descendant set is equivalent to the descendant set. Hence the "best" column and the "best" row folding candidates and patterns can be found by a procedure similar to the one described above. Let ρ^c (ρ^r) be the related upper bounds on the number of relations induced in G_R (G_C) by a column (row) fold. A column (row) fold is attempted if :

$$\alpha * \rho^c < \beta * \rho^r$$

$$(\alpha * \rho^c \geq \beta * \rho^r)$$

where $\alpha = \dfrac{C(\,O\,)-1}{C(\,O)}$ and $\beta = \dfrac{R(\,O\,)-1}{R(\,O)}$ are dynamic weighting factors which take into account the relative area saving achieved by a column (row) fold at that step of the algorithm and $C(0)$ $(R(0))$ is the column (row) cardinality.

It is important to remark that this strategy allows to achieve more folds in comparison with other algorithms performing column (row) folding after row (column) folding. Nevertheless it is straight-forward to constrain the selection so that all column (row) folds are tried first, if desired.

5. MULTIPLE CONSTRAINED FOLDING

As stated in Sections 1 and 2 the PLA constrained folding problems are related to the interconnection of the array to the outside circuitry. We classify the constraints on folding into two major categories:

1) Architectural or primary constraints

2) Secondary constraints.

Architectural constraints are related to the structure of the array and to the positions of input/output busses relative to the array. Secondary constraints are related to the positions of input and output lines inside the busses. Examples of architecture constrained folding problems are:

1A) Simple column folding with a subset of inputs and/or outputs connected to the top (bottom) of the array.

1B) Simple row folding with AND-OR-AND or OR-AND-OR architecture.

1C) Segmented arrays: the column set is partitioned into subsets, each forming a segment of the array. Columns are folded with columns in the same segment only and the sequence of segments is preserved.

The following folding problems involve secondary constraints:

2A) Column folding with bounded product-row assignment.

2B) Row folding with bounded column assignment.

2C) Column folding with bounded connection-row assignment.

2D) Column folding with ordered connection-row assignment.

The Master Algorithm presented in Section 4 can handle both architectural and secondary constraints. Different strategies are used in the two cases. To satisfy architectural constraints it is sufficient that folding candidates satisfy the following requirements for the related problems:

1A) *Columns connected to I/O busses on the top (bottom) of the array are folded either on top (bottom) of an unfolded column or folding list or not folded at all.*

1B) AND-OR-AND (OR-AND-OR) architecture. *Rows connected to input (output) columns that are connected to rows folded on the left or on the right are selected as candidates to be split on the left or on the right of the array respectively.*

1C) Selected candidates for column folding are constrained to be in the same segment. In the case of no more than three segments and simple row folding, the selection of candidates for row folding is as follows: *rows connected to columns in the leftmost (rightmost) segment are folded on the left (right) only or not folded at all.*

Unfortunately we cannot be sure that secondary constraints are satisfied only on the basis of an appropriate selection of folding candidates. The reason is that secondary constraints are related to the row (column) positions induced by a column (row) folding. Therefore, we present in this section two assignment algorithms that assign positions to rows and/or columns and checks if the secondary constraints are satisfied. We will present first the assignment algorithm for problem 2A . From this, an algorithm for problem 2B can be easily derived by interchanging rows with columns. Problems 2C and 2D are solved by a double assignment algorithm, based on the assignment algorithm of problem 2A.

5.1 Column folding with bounded product-row assignment

We consider in this section the problem of constraining product-term row positions only. We therefore refer to product-term rows as rows throughout this section.

We define **lower (upper) row bound map**: a map

$$L_R:\{r_i;\ i = 1, 2, \cdots, nr\} \rightarrow \{1, 2, \cdots, nr\}$$

$$(U_R:\{r_i;\ i = 1, 2, \cdots, nr\} \rightarrow \{1, 2, \cdots, nr\})$$

relating each row to a lower (upper) position bound.

We define **row assignment** $P:\{r_i;\ i = 1, 2, \cdots, nr\} \rightarrow \{1, 2, \cdots, nr\}$ a permutation of the rows and **implementable row assignment** a permutation compatible with an ordered column-folding set O; i.e. $P(r_x) < P(r_y)$ $\forall r_x < r_y \in QR^+(O)$

An **implementable bounded row assignment** is an implementable row assignment such that

$$L_R(r_j) \leq P(r_j) \leq U_R(r_j) \quad \forall\ j = 1, 2, \cdots, nr$$

Example 5.1.1 : For the logic array shown in fig. 2.1, the following lower and upper bounds are given:

$$L_R = 1, 1, 1, 4, 4, 6$$

$$U_R = 1, 3, 3, 6, 6, 6\ .$$

This means that r_1 is constrained to the first position, r_2 and r_3 are constrained between position 1 and 3 , and so on. The implementable row assignment ($r_1, r_4, r_2, r_3, r_5, r_6$) induced by the column folding shown in fig. 2.2 does not satisfy the given bound maps. On the contrary, the folded PLA shown in fig. 5.1 has the following implementable row assignment: ($r_1, r_2, r_3, r_5, r_4, r_6$). Note that rows are numbered from the top to the bottom of the array. ■

The optimal bounded row column folding problem can be stated as follows:

Find an implementable ordered column-folding set and a related implementable bounded row assignment that minimizes the column cardinality of the folded PLA.

Let us consider a graph interpretation of the following subproblem:

Given an ordered column-folding set and a lower and upper row bound maps, find an implementable bounded row assignment, if it exists.

The graph interpretation is useful to understand the underlying structure and to develop an algorithm and related heuristics. We associate to this subproblem a directed graph $G(R, N, A)$, with two node sets N and R, and a set of directed edges A.

The node sets R and N are in one to one correspondence with the row set and the set of the first nr natural numbers representing the possible row positions. Our problem consists in finding a matching between R and N, i.e. coupling each row-node to a position-node, so that all the required bounds are satisfied. We represent position bounds by a set of directed edges :

$$A = A_1 \cup A_2 \cup A_3 \cup A_4 \cup A_5$$

where : $A_1 \equiv \{(n_j, n_{j+1});\ j = 1, 2, \cdots, n-1\}$ represents the order on the sequence of the first nr natural numbers; $A_2 \equiv \{(n_i, r_j) | L(r_j) = i+1,\ \ j = 1, 2, \cdots, nr\}$ and $A_3 \equiv \{(r_j, n_i) | U(r_j) = i-1,\ \ j = 1, 2, \cdots, nr\}$ take into account the lower and upper bound maps ; $A_4 \equiv \{(r_i, r_j) | r_i < r_j \in QR(O)\}$ represents the order relations among the rows induced by the column folding.

Example 5.1.2 : Fig. 5.2a shows graph $G(R, N, A')$ $A' = A_1 \cup A_2 \cup A_3 \cup A_4$ for the PLA of fig. 2.1, the row bounds of example 5.1.1 and the ordered folding set $O = \{(c_7, c_8), (c_3, c_4), (c_2, c_5)\}$. ∎

Note that an edge from a node in N (R) to a node in R (N) represents now a strict lower (upper) bound. If a lower (upper) bound on a row position is 1 (nr), it can be represented by appending nodes n_0 (n_{nr+1}) to set N and by adding appropriate directed edges to A.

Moreover note that if a row, say $\tilde{r}$, has the position w as strict upper bound (i.e. $(\tilde{r}, n_w) \in A_3$) and must follow another row, say $\hat{r}$ (i.e. $(\hat{r}, \tilde{r}) \in A_4$), then row $\hat{r}$ has as strict upper bound a position lower or equal to $w-1$.

Example 5.1.3 : Row r_1 must be above r_2 which in turn must be above r_4. Since r_4 is required to be assigned to a position lower or equal to 6, r_1 must be assigned to a position lower or equal to 4. (In this case r_1 has already the more stringent constraint to be in position 1). ∎

We therefore define: $A_5 \equiv \{(r_k, n_{i-l}) | \exists r_j$ such that $(r_j, n_i) \in A_3$ and $\exists$ $l+1$ distinct nodes in R along the directed paths in A_4 from r_k to $r_j\}$. Similar considerations apply to lower bounds, but the assignment algorithm does not require that the set of directed edges is further increased.

Example 5.1.4 : The edges in subset A_5 are represented by dashed lines in fig. 5.2b. ∎

Our problem is to find an additional set of undirected edges E matching every node in R to one and only one node in N so that the resulting mixed graph $G(R, N, E, A)$ is acyclic.

Remark 5.1 : Column folding with bounded row assignment is equivalent to the sequencing problem with release times and deadlines where all task length are equal to one [17][18] and where a partial order on the tasks is given. ∎

The following algorithm will either construct a set of undirected edges such that graph $G(R, N, E, A)$ is acyclic or will return a flag if no possible edge set exists. We recall that the in-degree of a node is the number of directed edges incident to that node and the deletion of a node from a graph corresponds to remove the node from the node set and all edges incident to/from it from the edge set. The algorithm is described in Pidgin C.

```
ASSIGNMENT ALGORITHM
$E = \phi$ ;
delete $n_0$ from graph $G$;
for ( $i = 1 ; i \leq nr ; i = i + 1$ ) {
        if ( in-degree ($n_i$) $\neq 0$ ) return ( FALSE ) ;
        $Q = \{ r \in R$ ; in-degree $(r) = 0 \}$;
        if ( $Q = \phi$ ) return ( FALSE ) ;
        $r_j = r \in Q$ such that $(r_j, n_k) \in A$ and $k$ is minimal;
        $E = E \cup \{ n_i, r_j \}$ ;
        delete $n_i$ from graph $G$;
        delete $r_j$ from graph $G$;
}
return ( TRUE ) ;
```

The algorithm runs in linear time since it cycles at most nr times through the main loop. The algorithm uses a greedy strategy: at each iteration it matches the available position with lowest index to the most constrained node in R (i.e. selects the product-row with lowest upper bound). The algorithm finds an implementable bounded row assignment, if one exists, as proven by the following theorem.

Theorem 5.1 : The Assignment Algorithm returns " true " if and only if there exists a matching E such that graph $G(R, N, E, A)$ is acyclic.

The proof is reported in the appendix.

Example 5.1.5 : Consider the column folded logic array shown in fig. 5.1, and the related graph $G(R, N, A)$ shown in fig. 5.2. The implementable bounded-row assignments given by the algorithm is $(r_1, r_2, r_3, r_5, r_4, r_6)$.

The Assignment Algorithm replaces Step 3 of the Master Algorithm for column folding with bounded row assignment.

A different strategy for folding candidate selection is used. Since folding is limited by row positions, we try to fold columns incident to rows constrained to be in the top part of the array with columns incident to rows constrained to be in the bottom part of the array. We therefore can compute two "induced bound" maps for each column:

$$\tilde{L}(c_j) = \min_{r \in R(c_j)} L_R(r) \qquad j = 1, 2, \cdots, nc.$$

$$\tilde{U}(c_j) = \max_{r \in R(c_j)} U_R(r) \qquad j = 1, 2, \cdots, nc.$$

The column with the lowest (highest) entry in $\tilde{U}$ $(\tilde{L})$ is the most constrained to be folded on the top (bottom).

Example 5.1.6 : For the logic array of fig. 2.1 and the row bound maps of example 5.1, the induced bound maps are the following:

$$\tilde{L} = 1, 1, 1, 1, 4, 1, 1, 1, 4, 1$$

$$\tilde{U} = 6, 3, 1, 3, 6, 6, 1, 6, 6, 6$$

Hence columns c_3 and c_7 are the most constrained to be folded on the top part of the array and c_5 and c_9 on the bottom. ■

Hence a "good" selection is the candidate pair (c_i, c_k) such that

$$c_i = arg_{j=1,2,\ldots,nc}^{\min} \; \tilde{U}(c_j)$$

$$c_k = arg_{j=1,2,\ldots,nc}^{\max} \; \tilde{L}(c_j)$$

A more considerate choice takes also care of the number of ancestor-descendant relations induced in the mixed graph, as shown in Section 4. Therefore we use weighted selection criterion:

$$c_i = arg_{j=1,2,\ldots,nc}^{\min}[\alpha|\tilde{D}(v_j)| + \beta\tilde{U}(v_j)]$$

$$c_k = arg_{j=1,2,\ldots,nc}^{\min}[\alpha|A(v_j)| - \beta\tilde{L}(v_j)]$$

Example 5.1.7 : The first folding pair selected by the algorithm is (c_7, c_9). ■

Similar considerations apply, "mutatis mutandis", to the multiple folding candidate selection.

Remark 5.2 : The graph interpretation and an algorithm for the row folding with bounded column assignment problem can be derived "mutatis mutandis" from this problem. ■

5.2 Column folding with bounded connection-row assignment

We refer in this section to a logic array implemented with connection-rows for routing input and output signals as described in Section 2. According to these architectures, there are two sets of connections rows contacting the columns of the left and right array respectively. For the sake of simplicity, we will consider constrained folding of one array only.

Both proposed architectures support at most as many connection-rows as product-rows. Since each column is contacted to a connection row, we require throughout the section that the number of columns in the considered array is at most equal to the number of rows. Most PLA satisfy this assumption.

We define **connection-row assignment** a one-to-one map: $T:\{c_i,\ i = 1, 2, \cdots, nc\} \rightarrow M \subseteq \{1, 2, \cdots, nr\}$ such that $j = T(c_i)$ if column c_i is contacted to the connection row in the j^{th} position.

Example 5.2.1 : Consider the OR plane of the PLA shown in fig. 2.1. Fig. 5.3 shows the unfolded array with the connection row assignment:
$T(c_7) = 1 \quad T(c_8) = 2 \quad T(c_9) = 5 \quad T(c_{10}) = 6$. ■

We define **physical connection-row** set M the image of T. Its elements are the position of the connection-rows which are physically implemented. Note that there are $\Delta = nr - nc$ slack connection-rows which are not implemented and whose positions are irrelevant to the problem.

We define **lower (upper) connection-row bound map** a map:

$$L_C : \{c_i, \ i = 1, 2, \cdots, nc\} \rightarrow 1, 2, \cdots, nr$$

$$(U_R : \{c_i, \ i = 1, 2, \cdots, nc\} \rightarrow 1, 2, \cdots, nr)$$

relating each column to a lower (upper) position bound on the position of the contacted connection-row.

Example 5.2.2 : For the OR plane of the PLA shown in fig. 2.1 , the following bounds are given:

$$L_C = 1, 1, 4, 6$$

$$U_C = 1, 3, 6, 6$$

This means that the first column of the OR plane (c_7) must be connected to a connection-row in position 1 ; the second one (c_8) to a connection-row whose position is bounded between 1 and 3 ; and so on. ■

An **implementable connection-row assignment** is an assignment compatible with a column ordered folding set, i.e. is an assignment such that :

$$\max(P(R(c_{i,j-1}))) < T(c_{i,j}) < \min(P(R(c_{i,j+1}))) \ j = 1, 2, \cdots, n$$

$\forall$ column $c_{i,j}$ in folding list o_i with cardinality n, where by definition:

$$\max(P(R(c_{i,0}))) = 0 \quad \text{and} \quad \max(P(R(c_{i,n+1}))) = \infty$$

Example 5.2.3 : Consider the folded OR plane shown in fig. 2.2 with the ordered folding set $O = \{(c_7, c_9), (c_8, c_{10})\}$. An implementable connection-row assignment is:

$$T(c_7) = 1 \quad T(c_8) = 2 \quad T(c_9) = 3 \quad T(c_{10}) = 6$$

The connection-row contacted to c_8 is in position 2, and therefore is above (has lower index than) the product rows connected to c_{10} (in positions 4 and 6). The connection row contacted to c_{10} is in position 6 and is below (follows) the product rows connected to c_8 (in positions 3 and 2). ■

An **implementable bounded connection-row** assignment is an implementable connection-row assignment such that :

$$L_C(c_j) \leqslant T(c_j) \leqslant U_C(c_j) \quad j = 1, 2, \cdots, nc$$

Example 5.2.4 : The implementable connection row-assignment of example 5.2.3 does not satisfy the bounds given in example 5.2.2. An implementable bounded connection row-assignment is:

$$T(c_7) = 1 \quad T(c_8) = 2 \quad T(c_9) = 4 \quad T(c_{10}) = 6$$

Fig. 5.4 shows a folded implementation of the OR plane compatible with the bounded connection-row assignment. ■

We can now state the column folding with bounded connection-row assignment problem as follows:

Find an implementable ordered column-folding set and a related implementable bounded connection-row assignment which minimizes the column cardinality of the folded PLA.

As we did for the previous problem, we consider a graph interpretation of the following subproblem:

Given an ordered column-folding set and a lower and upper connection-row bound maps, find an implementable bounded connection-row assignment, if it exists.

Note that an implementable bounded connection row assignment requires, by definition, a product row assignment, because the positions of rows in both sets influence each other. Hence the problem consists in finding the two row assignments compatible with the ordered column-folding set, if they exist.

We associate to this subproblem a directed graph $G(R, N, C, A)$, with three node sets R, N and C and a directed set of edges A. The node sets R, C and N are in one to one correspondence with the row set, the column set and the set of the first nr natural numbers respectively.

We represent the bounds on the row positions by a set of directed edges:

$$A = A_1 \cup A_2 \cup A_3 \cup A_4 \cup A_5 \cup A_6 \cup A_7 \cup A_8$$

where A_1 and A_4 are defined as in section 5.1, $A_2 = \{(n_i, c_j) | L_C(c_j) = i+1 ; \ j = 1, 2, \cdots, nc\}$ and $A_3 = \{(c_j, n_i) | U_C(c_j) = i-1 ; \ j = 1, 2, \cdots, nc\}$ take into account the lower and upper bound maps.

Example 5.2.5 : Fig. 5.5a shows graph $G(R, N, , C, A')$, $A' = A_1 \cup A_2 \cup A_3 \cup A_4$ in the case that the OR plane of the PLA of fig. 2.1 is folded and the ordered column-folding set is: $O = \{(c_7, c_9), (c_8, c_{10})\}$ and is compatible with the bounds given in Example 5.2.2.

■

We consider the mutual relations among products and connection-rows by the edge subsets: $A_6 = \{(\tilde{r}, \hat{c}) | \tilde{r} \in R(\tilde{c})$ and $\tilde{c}$ is split on top of $\hat{c}\}$ and $A_7 = \{(\hat{c}, \tilde{r}) | \tilde{r} \in R(\tilde{c})$and $\hat{c}$ is split on top of $\tilde{c}\}$. In words , if column $\tilde{c}$ is folded on top of $\hat{c}$, then all the rows (product and connection) connected to $\tilde{c}$ must be assigned to positions with index lower than the positions of all the rows connected to $\hat{c}$.

Example 5.2.6 : Fig. 5.5b shows the edges in subsets A_6 and A_7 for the problem of example 5.2.5.

■

Moreover note that if a column , say $\hat{c}$, has as strict upper bound the position w (i.e. $(\hat{c}, n_w) \in A_3$, $(\tilde{r}, \hat{c}) \in A_6$ and $(r, \tilde{r}) \in A_4$, then r has as upper bound the position $w-2$. We therefore define: $A_5 \equiv \{(r_k, n_{i-l}) | \exists r_j, \exists \hat{c}$ such that $(r_j, \hat{c}) \in A_6$, $(\hat{c}, n_i) \in A_3$ and $\exists l > 0$ distinct nodes in R along the directed paths from r_k to r_j}. The edges in this set represent the upper bounds on the position of each product-row induced by folding. Note that all nodes in R must be assigned to a position lower than $nr+1$. Hence we append to A_5 the edges (r_k, n_{nr+1}) $\forall r_k \in R$ having no explicit upper bound .

Example 5.2.7 : Fig. 5.5c shows the edges in subset A_5 for the problem of example 5.2.5. ■

Similarly , upper bounds induced on the column positions are represented by: $A_8 \equiv \{(c_k, n_{i-1}) | \exists (c_k, r_j) \in A_7$ and $(r_j, n_i) \in A_5\}$.

Example 5.2.8 : Fig. 5.5d shows the edges in subset A_8 for the problem of example 5.2.5. ■

In graph terms, this problem is to find a set of undirected edges E matching every node in R and in C to one and only one node in N so that the resulting mixed graph $G(R, N, C, E, A)$ is acyclic. Note that in general the number of columns and hence of physical connection-rows required is smaller than the number of rows by Δ and we take advantage of this in the double assignment algorithm.

DOUBLE ASSIGNMENT ALGORITHM

```
$E = \phi$;
$\Delta = nr - nc$;
delete $n_0$ from graph $G$;

for ( $i = 1$ ; $i \leq nr$ ; $i = i + 1$ ) {

        if ( in-degree ( $n_i$ ) = 0 ) return ( FALSE );

        $Q = \{r \in R$ ; in-degree ( $r$ ) = 0$\}$;

        if ( $Q = \phi$ ) return ( FALSE ) ;

        $r_j = r \in Q$ such that $(r_j, n_k) \in A$ and $k$ is minimal;

        $E = E \cup (n_i, r_j)$;

        $H = \{c \in C$ ; in-degree ( $c$ ) = 0$\}$;

        if ( $H = \phi$ ) {

                $\Delta = \Delta - 1$;
                if ( $\Delta < 0$ ) return ( FALSE ) ;

        }
        else {
                $c_l = c \in H$ such that $(c_l, n_k) \in A$ and $k$ is minimal ;

                $E = E \cup (n_i, c_l)$;

                delete $c_l$ from graph $G$;
        }

        delete $r_j$ from graph $G$;
        delete $n_i$ from graph $G$;

}

return ( TRUE ) ;
```
■

The double assignment algorithm runs in linear time and uses a greedy strategy. At each iteration, it tries to match the available position with lowest index with the most constrained product and connection-rows. Note that a connection-row need not be assigned at each iteration, but the total number of slack positions must be lower or at least equal to Δ.

Theorem 5.2: The assignment algorithm returns "true" if and only if there exists a set of undirected edges E matching each node in R and in C to one and only one node in N such that $G(R, N, C, E, A)$ is acyclic.

The proof is reported in the appendix

The double assignment algorithm replaces Step 3 of the Master Algorithm for column folding with bounded connection-row assignment

The selection of folding candidates is based on the following strategy. Try to fold columns incident to connection-rows constrained to be in the top part of the array with columns connected to connection-rows constrained to be in the bottom part of the array. Therefore the candidate selection follows the outlines presented in section 5.1, where $\tilde{L}(c_j) = L(c_j)$ and $\tilde{U}(c_j) = U(c_j)$. Also in this case, a considerate choice of folding candidates uses a selection criterion weighting the number of ancestor-descendant relations induced by the fold and the required row positions in the array.

5.3 Column folding with ordered connection-row assignment

We extend to this section the considerations on multiple column folded PLA implementation and the basic definitions presented in Section 5.2.

We define **order map** $S:\{c_i;\ i = 1, 2, \cdots, nc\} \to \{1, 2, \cdots, nc\}$ a one to one map relating each column to the required relative position of the contacted connection-row. We define **implementable ordered connection-row assignment** an implementable connection-row assignment such that :

$$T(c_i) < T(c_j) \quad \text{if} \quad S(c_i) < S(c_j) \quad \forall i,\ j = 1, 2, \cdots, nc$$

Example 5.3.1 : Consider the OR plane of the PLA shown in fig. 2.1 and the following order map:

$$S(c_7) = 2 \quad S(c_8) = 1 \quad S(c_9) = 3 \quad S(c_{10}) = 4$$

This means that column folding is constrained so that the connection-row to c_8 is in the topmost position, followed by those connecting c_7, c_9 and c_{10} in the order. Fig. 5.6 shows a folded implementation with the implementable ordered connection-row assignment: $T(c_7) = 2$ $T(c_8) = 1$ $T(c_9) = 3$ $T(c_{10}) = 4$. ∎

We state the column folding with ordered connection-row assignment problem as follows:

Find an implementable ordered column-folding set and a related implementable ordered connection-row assignment, which minimizes the column cardinality of the folded PLA.

This problem is equivalent to column folding with the following bounds on connection-row positions:

$$L_C(c_i) = S(c_i) \quad \forall i = 1, 2, \cdots, nc$$

$$U_C(c_i) = S(c_i) + \Delta \quad \forall i = 1, 2, \cdots, nc$$

with the additional constraint on the order of the connection-rows.

As we did in the previous section, we give a graph representation for a subproblem:

Given an ordered column-folding set and an order map, find an implementable ordered connection-row assignment, if it exists.

The graph representation of this subproblem is given by graph $G(R, N, C, A)$ introduced in Section 5.2 where an additional subset of directed edges is added to take care of the order map:

$$A_9 \equiv \{(c_i, c_j) | i = S(c_k), j = S(c_{k+1}), k = 1, 2, \cdots, nc - 1\}$$

The Double Assignment Algorithm can be used to replace Step 3 of the Master Algorithm for the column folding with ordered connection-row assignment problem.

Example 5.3.2 : Fig. 5.7 shows graph $G(R, N, C, A)$ for the order map of example 5:3.1 and the ordered folding set $O = \{(c_8, c_9)\}$ ■

Remark 5.3 : In the case that there are no slack positions or in the case that we are not interested in taking advantage of the slack positions, the column-folding with ordered connection-row assignment problem can be solved more easily by the following equivalent formulation: ***column folding with bounded product-row assignment**, where bounds on row positions are dynamically induced by column-folding*. In particular:

$$U_R(c_{i,j}) = S(c_{i,j+1}) + \delta - 1$$

$$L_R(c_{i,j+1}) = S(c_{i,j}) + \delta + 1$$

$$\forall c_{i,j} \in o_i \ , \ \forall o_i \in O \text{ and } \textit{any fixed } \delta \text{ s.t. } 0 \leq \delta \leq \Delta$$

An implementable product-row assignment satisfying the above bounds is a necessary and sufficient condition for the existence of the implementable ordered connection-row assignment $T(c_j) = S(c_j) + \delta$. ■

The selection of folding candidates is based on the following strategy. Try to fold columns incident to connection-rows constrained to be in the top part of the array with columns connected to connection-rows constrained to be in the bottom part of the array. Therefore the candidate selection follows the outlines presented in section 5.1, where now: $\tilde{L}(c_j) = S(c_j)$ and $\tilde{U}(c_j) = S(c_j)$. Also in this case, a considerate choice of folding candidates uses a selection criterion weighting the number of ancestor-descendant relations induced by the fold and the required row positions in the array.

6. PLEASURE

PLEASURE is an interactive program for simple/multiple constrained/unconstrained row and/or column folding of Programmable Logic Arrays.

The PLA description is given as input to the program in the form of two-level sum-of-products logical implicants.

The output of the program is a symbolic table representing the folded array with the positions of the active devices corresponding to the cares of the logic function, the locations of the cuts and the contacts between columns ad connection rows. The symbolic table is suitable to be processed by a *silicon assembler* program which generates the mask layout of the array according to a given technology. Note that the symbolic table generated by PLEASURE is technology independent.

The program is a command interpreter: input files can be read and edited; logic arrays can be folded in a single run or one fold at a time. This allows the user to monitor PLA folding step by step, by displaying the partially folded array. The user can enter column and/or row folding candidates of his choice and verify the implementability of his selection. When PLAs are folded in a single run a soft interrupt capability allows the user to halt the compaction at any point to see the partially compacted array before resuming folding execution. The program can be run in a silent mode (i.e. with no interaction with the user), so that it can be interfaced with other programs in a system for automated synthesis of PLA's.

Folding instructions are entered to the program along with the PLA description in the input file. PLEASURE allows column (row) folding only and row and column folding.

Column folding can be simple or multiple, constrained or unconstrained in either or in both planes. Architectural constraints can be set on column positions. Columns can be required to be folded on the top (bottom) of the array or not folded at all. Column folded arrays can be segmented into three adjacent planes, so that columns in the external planes, can be multiple folded and contacted by connection rows. Secondary constraints can be put on product ad connection row positions. In particular column folding with bounded or order connection row assignment can be achieved.

Row folding can be simple or multiple. Simple row folded arrays can be constrained to have an AND-OR-AND or OR-AND-OR architecture. Secondary constrained can be put on the column positions.

Row (column) folding can follow column (row) folding. Row folds can be alternated with column folds, by allowing the program to choose the local "best" fold at each step. This procedure achieves the best results as far as compaction is concerned. Multiple row and column folded PLA can be constrained by input/output position. An input (output) can be required to be connected to the top, bottom, left or right of the array.

Program PLEASURE is coded in *ratfor* and consists of about 5000 lines. Intermediate code in *fortran*77 is available. PLEASURE runs in a VAX-UNIX® environment, but is easily transportable to other machines.

Some PLEASURE output files are reported in fig. 6.1a and 6.1b for the PLA of Fig. 2.1 and different folding requirements. PLEASURE has been tested on a large set of industrial arrays. To compare results obtained with arrays of different sizes, the following foldings have been tried: i) unconstrained folding; ii) column folding with constrained row positions: $L(r_i) = max(i-\alpha, 0); U(r_i) = min(i+\alpha, nr); \alpha = \frac{nr}{10}$; iii) column folding with constrained connection-row positions: $L_C(c_i) = max(i-\alpha, 0); U_C(c_i) = min(i+\alpha, nr); \alpha = \frac{nr}{10}$; iv) column folding with ordered connection-row assignment: $S(c_i) = i, i = 1, 2, \cdots, nc$. The folding results and execution time on a VAX 11/780 computer are reported in table 1.

7. CONCLUSIONS

In this paper we addressed the multiple constrained folding problem of Programmable Logic Arrays. A heuristic algorithm for multiple folding has been presented as well as two assignment algorithms for PLA row/column constrained positioning. A computer program, PLEASURE, has been described and shown to be an effective tool for interactive topological design of logic arrays.

The PLEASURE output file contains all the topological informations for the implementation of multiple folded arrays. The layout of the masks of the folded array can be obtained from the PLEASURE output file by means of a *silicon assembler* program, once an implementation technology is chosen. We have presented two PLA structures which support multiple folded arrays in MOS technology : the former uses two levels of metal (poly) and the latter one level of metal and poly.

Future work include the development of a *silicon assembler* program, that can generate the multiple folded PLA mask layout according to different architectures and design rules.

PLEASURE is a part of the integrated system for Programmable Logic Arrays and Finite State Machines automated design developed at the University of California,Berkeley.

8. REFERENCES

[1] H.Fleisher and L.I.Maissel "An Introduction to Array Logic" *IBM Jour. on Res. and Devel.*, vol 19, pp.98-109, Mar 1975

[2] M.S.Schmookler "Design of Large ALUs Using Multiple PLA Macros" *IBM Jour. on Res. and Devel.*, vol.24 pp.2-14 Jan 1980

[3] S.J.Hong,R.G.Cain and D.L.Ostapko "MINI:a Heuristic Approach for Logic Minimization" *IBM Jour. on Res. and Devel.*, vol 18, pp 443-458, Sep 1974

[4] R.Brayton,G.D.Hachtel,L.Hemachanandra,A.R.Newton and A.L.Sangiovanni Vincentelli "A Comparison of Logic Minimization Strategies Using Espresso. An APL Program Package for Partitioned Logic Minimalization" *Proc. Int. Symp. on Circ. and Syst.* pp. 42-48, Rome 1982

[5] G.D.Hachtel,A.R.Newton and A.L.Sangiovanni Vincentelli "An Algorithm for Optimal PLA Folding" *IEEE Trans on CAD of Int. Circ. and Syst.*, vol 1, No 2, Apr 1982

[6] R.A.Wood "A High Density Programmable Logic Array Chip", *IEEE Trans. Comput.*, vol C-28, pp. 602-608, Sep 1979

[7] G.D.Hachtel,A.R.Newton and A.L.Sangiovanni Vincentelli "An Algorithm for Optimal PLA Folding" *Proc. Int. Circ. and Comp. Conf.*, New York,N.Y. Oct 1980

[8] D.L.Greer " An Associative Logic Matrix " *IEEE jour. of Solid State Circ.*, vol SC-11, No 5, pp 679-691 Oct 1976

[9] J.F.Paillotin " Optimization of the PLA Area" *Proc. 18th Des. Autom. Conf.* , pp 406-410, Nashville Jun 1981

[10] S.Chuquillanqui and T. Perez Segovia " PAOLA: A Tool for Topological Optimization of Large PLAs" *Proc. 19th Des. Autom. Conf.*, pp 300-306, Las Vegas Jun 1982

[11] G.D.Hachtel,A.R.Newton and A.L.Sangiovanni Vincentelli "Techniques for Programmable Logic Arrays Folding" *Proc. 19th Des. Autom. Conf.*, pp 147-152, Las Vegas, Jun 1982

[12] C.Mead and L.Conway "*Introduction to VLSI Systems*" Addison Wesley 1980

[13] G. De Micheli " Pleasure: A Program for constrained PLA Folding" *Internal Report*, Harris Corporation , 1980

[14] I.Suwa and W.J.Kubitz " A Computer Aided Design System for Sement-Folded PLA Macro cells" *Proc. 18th Des. Autom. Conf.* pp 398-405, Nashville, Jun 1981

[15] A.V.Aho J.E.Hopcroft and J.D.Ullman "*The Design and Analysis of Computer Algorithms*" Addison Wesley 1974

[16] M.Luby U.Vazirani V. Vazirani and A. Sangiovanni-Vincentelli "Some Theoretical Results on the Optimal PLA Folding Problem" *Proc. Int. Circ. and Comp. Conf.*, pp 165-170, New York,N.Y., Oct 1982

[17] M.R.Garey and D.S.Johnson "*Computers and Intractability*" W.H.Freeman and Company San Francisco

[18] E.L.Lawler "Optimal Seqencing of a Single Machine Subject to Precedence Constraints" *Management Science* vol 19 No 5, pp. 544-546, Jan 1973

APPENDIX

Proof of Theorem 3.1:

(if)

Assume that the folding set is implementable. For the sake of contradiction, suppose that $\exists$ a χ*-cycle* in $G(V, E, A(O))$. Without loss of generality, we can label the vertices of the cycle so that :

$$\{v_{p-1}, v_p\} \in E.$$

$$\{v_l, v_{l+1}\} \in E.$$

$$(v_k, v_{k+1}) \in A \quad \forall k \in \{1, 2, \cdots, l-1\} \cup \{l+1, l+2, \cdots, m-1\}$$

It is always possible to achieve such a labeling, because a χ*-cycle* has at least two undirected edges ($\{v_{p-1}, v_p\}$ and $\{v_l, v_{l+1}\}$) and two paths of directed edges (joining v_1 to v_l and v_{l+1} to v_{m-1}). Moreover a path of directed and undirected edges (possibly of zero length) joins v_m to v_{p-1}.
Since paths of directed edges are related to column ordered folding lists, and the column ordered folding lists induce a row order relation, we have :

$$R(c_1) < R(c_l);$$

and :

$$R(c_{l+1}) < R(c_m).$$

Take any row $r \in R(c_l) \cap R(c_{l+1})$:

$$R(c_1) < r < R(c_m)$$

and from the definition of transitive closure :

$$\{R(c_1) < R(c_m)\} \subset QR^+(O)$$

Since there is a finite number of vertices along the χ*-cycle* from vertex v_{l+1} to vertex v_{p-1}, by repeating the same argument and by the transitivity of $QR^+(O)$:

$$\{R(c_{l+1}) < R(c_{p-1})\} \subset QR^+(O).$$

Let row $\hat{r} \in R(c_{p-1}) \cap R(c_p)$. From the transitive closure relation:

$$R(c_{l+1}) < \hat{r}$$

and in particular:

$$r < \hat{r}$$

But since $c_1 = c_p$ and $R(c_1) < \hat{r}$, then :

$$\hat{r} < r$$

Hence we have a contradiction because $QR^+(O)$ is not a partial order on the set Z^+.

(only if)

Assume that $G(V, E, A(O))$ has no χ-cycles. For the sake of contradiction suppose that $QR^+(O)$ is not a partial order. Therefore there exist two rows, r_1 and r_1, such that:

$$r_1 <^+ \tilde{r}_1$$

and:

$$\tilde{r}_1 <^+ r_1$$

Hence there exists a sequence:

$$[r_1, r_2, \cdots, r_n, \tilde{r}_1]$$

such that:

$$r_j \in R(c_{1,j}) \cap R(c_{2,j}) \quad j = 1, 2, \cdots, n$$

where: $c_{1,1} = c_{2,1}$, $c_{1,n} = c_{2,n}$, $(c_{2,j-1}, c_{1,j}) \in A$ $j = 2, 3, \cdots, n$ and either $c_{1,j} = c_{2,j}$ or $\{c_{1,j}, c_{2,j}\} \in E$, $j = 2, 3, \cdots, n-1$. Hence there is a directed path from $c_{1,1}$ to $c_{2,n}$ having no more than one consecutive undirected edge. Moreover $r_1 \in R(c_1)$and$(c_{2,n}, c_1) \in A$.
Furthermore there exist a sequence:

$$[\tilde{r}_1, \tilde{r}_2, \cdots, \tilde{r}_n, r_1]$$

such that:

$$\tilde{r}_j \in R(\tilde{c}_{1,j}) \cap R(\tilde{c}_{2,j}) \quad j = 1, 2, \cdots, n$$

where $\tilde{c}_{1,1} = \tilde{c}_{2,1}$, $\tilde{c}_{1,n} = \tilde{c}_{2,n}$, $(\tilde{c}_{2,j-1}, \tilde{c}_{1,j}) \in A$ $j = 2, 3, \cdots, n$ and either $c_{1,j} = c_{2,j}$ or $\{c_{1,j}, c_{2,j}\} \in E$, $j = 2, 3, \cdots, n-1$. Hence there is a directed path from $c_{1,1}$ to $c_{2,n}$ having no more than one consecutive undirected edge. Moreover $r_1 \in R(c_1)$and$(c_{2,n}, c_1) \in A$, and either $c_{1,1}$ and c_1 ($c_{1,1}$ and c_1) coincide or $\{c_{1,1}, c_1\} \in E$ ($\{c_{1,1}, c_1\} \in E$).
Thus $[c_{1,1}, \cdots, c_{2,n}, c_1, c_{1,1}, \cdots, c_{2,n}, c_1, c_{1,1}]$ is a χ-cycle. Hence we have a contradiction.

Proof of Theorem 5.1:

(if)

Suppose that the algorithm returns " false " at step i ; i.e. after having matched $i-1$ row nodes to position nodes. For the sake of contradiction, suppose that there exists a matching $E' = \{ \{r'_j, n_j\}$, $j = 1, 2, \cdots, nr\}$, such that $G(R, N, E', A)$ is acyclic.
The algorithm returns "false" in one of the following two cases:

Case 1 : $Q = \phi$ at step i.

There are $nr - i + 1$ row nodes that must be matched to position nodes $n_j, j > i$. Since $|\{n_j \in N, j > i\}| = nr - i$, no row assignment can be found satisfying the given bounds. In fact, since $\exists j > i$ such that $(n_j, r'_i) \in A$, then $[n_i, \cdots, n_j, r'_i, n_i]$ is a cycle in $G(R, N, E', A)$. Therefore we have a contradiction.

Case 2 : in-degree$(n_i) \neq 0$ at step i.

Let E^p be the partial assignment constructed by the algorithm, i.e. $E^p = \{\{n_j, r_j^p\}, j = 1, 2, \cdots, i-1\}$.
We show first that the matching E' can be transformed into another matching E'', such that $G(R, N, E'', A)$ is acyclic and the row nodes matched to $n_j, j = 1, 2, \cdots, i-1$ in E^p and E'' are identical. For this reason let:

$$a = arg \min \{j \mid r'_j \neq r_j^p\}$$

Nodes r'_a and r_a^p have no incoming directed edges from $\{n_j, j \geq a\}$.
Moreover $\exists n_h, n_k \in N, k \geq h > a$, such that $(r'_a, n_k) \in A$ and $(r_a^p, n_h) \in A$. Let $n_b \in N$, s.t. $\{n_b, r_a^p\} \in E'$. Then $a < b < h \leq k$. Let us consider the matching:

$$E'' = E' \cup \{r'_a, n_b\} \cup \{r_a^p, n_a\} - \{r'_a, n_a\} - \{r_a^p, n_b\}$$

We claim that $G(R, N, E'', A)$ is acyclic. If not, there would be at least a directed path joining one of the following node pairs:
i) n_b, r'_a
ii) n_a, r_a^p
iii) r'_a, n_b
iv) r_a^p, n_a
and $G(R, N, E', A)$ would have a cycle. In fact:
i) Since $b > a$ and there is a directed path from n_a to n_b, there would be the cycle $[n_b, r'_a, n_a, \cdots, n_b]$.
ii) Since r'_a has no incoming directed edges from $n_j, j \geq a$ there would be a directed path from n_a to a node $n_j, j < a$ and therefore there would be the cycle $[n_a, \cdots, n_j, \cdots, n_a]$.
iii) Since r'_a has no directed edges into $n_j, j < h$, there would be a directed path from a node $n_j, j \geq h$ to n_b , and therefore there would be the cycle $[n_b, \cdots, n_j, \cdots, n_b]$.
iv) Since $b > a$ and there is a directed path from n_a to n_b, there would be the cycle $[r_a^p, n_a, \cdots, n_b, r_a^p]$.

Let now $\tilde{E}'' = \{\{n_j, r''_j\} \in E'', j = 1, 2, \cdots, i-1\}$. If $\tilde{E}'' = E^p$, then n_i has no incoming directed edges from $\{r''_j \in R \mid j > i\}$. Suppose that $\{(r''_k, n_i) \in A$ and $k > i$. Then $[r''_k, n_i, \cdots, n_k, r''_k]$ would be a cycle in $G(R, N, E'', A)$. We therefore have a contradiction. If $\tilde{E}'' \neq E^p$, then we can construct a finite sequence of matchings $E'', E''', \cdots, E^*$ using the procedure shown above , so that $G(R, N, E^*, A)$ is acyclic and $\tilde{E}^* = E^p$, where : $\tilde{E}^* = \{\{n_j, r_j^*\} \in E^*, j = 1, 2, \cdots, i-1\}$. Also in this case we have a contradiction.

(only if)

The algorithm terminates in a finite number of steps, because it attempts at most nr assignment. Let $E = \{ \{n_j, r_j\}, j = 1, 2, \cdots, nr\}$ be the assignment constructed by the algorithm. Since n_j and r_j have no incoming directed edges from $\{\{n_k \mid k>j\} \cup \{r_k \mid k>j\} \ \ j = 1, 2, \cdots, nr\}$ by construction, then $G(R, N, E, A)$ is acyclic.

Proof of Theorem 5.2 :

(if)

Suppose that the algorithm returns " false " at step i . For the sake of contradiction, suppose that there exists a matching E' such that $G(R, N, C, E', A)$ is acyclic. In particular: $E' = \{ \ \{r'_j, n_j\} \ , \ j = 1, 2, \cdots, nr\} \cup \{ \ \{c'_j, n_j\},$ $\forall j \in M' \subseteq \{1, 2, \cdots, nr\} \}$, where M' is the physical connection row set corresponding to the matching E'.
The algorithm returns "false" in one of the following three cases:

Case 1 : $Q = \phi$ at step i.

There are $nr - i + 1$ row nodes that must be matched to position nodes $n_j, j>i$. Since $|\{n_j \in N, j>i\}| = nr - i$, no row assignment can be found satisfying the given bounds. In fact, since $\exists j>i$ such that $(\ n_j, r'_i \ ) \in A$, then $[n_i, \cdots, n_j, r'_i, n_i]$ is a cycle in $G(R, N, C, E', A)$. We therefore have a contradiction.

Case 2 : $H = \phi$ and $\Delta<0$ at step i.

There are $nr - i + 1$ connection_row nodes that must be matched to position nodes $n_j, j>i$. Since $|\{n_j \in N, j>i\}| = nr - i$, no connection_row assignment can be found satisfying the given bounds. In fact, since $\exists j>i$ such that (n_j, c'_i) $\in A$, then $[n_i, \cdots, n_j, c'_i, n_i]$ is a cycle in $G(R, N, C, E', A)$. We therefore have a contradiction.

Case 3 : in-degree$(n_i) \neq 0$ at step i.

Let E^p be the partial assignment constructed by the algorithm.
We show first that the matching E' can be transformed into another matching E'', such that $G(R, N, C, E'', A)$ is acyclic and row and connection_row nodes matched to $n_j, j = 1, 2, \cdots, i-1$ in E^p and E'' are identical. For this reason let:

$$a = arg \ \min \ \{j \mid r'_j \neq r^p_j\}$$

$$d = arg \ \min \ \{j \mid c'_j \neq c^p_j \ \ or \ \ \{c'_j, n_j\} \notin E' \ \text{ and } \ \{c^p_j, n_j\} \in E^p\}$$

If ($a \leq d$) let:

$$E'' = E' \cup \{r'_a, n_b\} \cup \{r^p_a, n_a\} - \{r'_a, n_a\} - \{r^p_a, n_b\}$$

If ($d<a$) , $c'_d \neq c^p_d$ and $\{c'_d, n_d\} \in E'$ let:

$$E'' = E' \cup \{c'_a, n_e\} \cup \{c^p_a, n_d\} - \{c'_a, n_d\} - \{c^p_a, n_e\}$$

where $n_e \in N$ $s.t.$ $\{n_e, c^p_d\} \in E'$.

If ($d<a$) , $\{ c'_d, n_j\} \not\in E'$ and $\{ c^p_d, n_j \} \in E^p$ let:

$$E'' = E' \cup \{c^p_d, n_d\} - \{c^p_d, n_e\}$$

We can show with an argument similar to the one used in the proof of theorem 5.1, that graph $G(R, N, C, E'', A)$ is acyclic, because otherwise graph $G(R, N, C, E', A)$ would have a cycle and violate our assumption.

Let now $\tilde{E}'' \subseteq E''$ be the subset of the undirected edges having an end_point in $n_j, j = 1, 2, \cdots, i-1$. If $\tilde{E}'' = E^p$, then n_i has no incoming directed edges from $\{r''_j \in R \mid j>i\} \cup \{c''_j \in C \mid j>i\}$. Suppose that $\{(r''_k, n_i) \in A$ and $k>i$. Then $[r''_k, n_i, \cdots, n_k, r''_k]$ would be a cycle in $G(R, N, C, E'', A)$. Suppose that $\{(c''_k, n_i) \in A$ and $k>i$. Then $[c''_k, n_i, \cdots, n_k, c''_k]$ would be a cycle in $G(R, N, C, E'', A)$. We therefore have a contradiction.
If $\tilde{E}'' \neq E^p$, then we can construct a finite sequence of matchings $E'', E''', \cdots, E^*_{\sim}$ using the procedure shown above , so that $G(R, N, C, E^*, A)$ is acyclic and $\tilde{E}^* = E^p$, where : $\tilde{E}^* \subseteq E^*$ is the subset of the undirected edges having an end_point in $n_j, j = 1, 2, \cdots, i-1$. Also in this case we have a contradiction.

(only if)

The algorithm terminates in a finite number of steps, because it attempts at most $2 * nr$ assignment. Let E be the assignment constructed by the algorithm. Since n_j, r_j and c_j have no incoming directed edges from $\{\{n_k \mid k>j\} \cup \{r_k \mid k>j\} \cup \{c_k \mid k>j\} \ \ j = 1, 2, \cdots, nr \ \}$ by construction, then $G(R, N, C, E, A)$ is acyclic.

TABLE 1

Comparison of PLAs folded by PLEASURE with different constraints.					
PLA	size nr*(ni+no)	Constraints	Folding lists	Folded Area Unfolded Area = 100	Time (sec)
PLA 1	30*(8+31)	none	7	29	8
	30*(8+31)	row positions	14	51	14
	30*(8+31)	conn_row positions	15	53	23
	30*(8+31)	ordered conn_rows	15	53	16
PLA 2	52*(23+20)	none	7	37	15
	52*(23+20)	row positions	12	60	34
	52*(23+20)	conn_row positions	13	46	62
	52*(23+20)	ordered conn_rows	13	58	53
PLA 3	86*(8+63)	none	9	56	112
	86*(8+63)	row positions	15	67	257
	86*(8+63)	conn_row positions	12	63	305
	86*(8+63)	ordered conn_rows	15	73	328
PLA 4	62*(24+14)	none	11	58	23
	62*(24+14)	row positions	10	73	36
	62*(24+14)	conn_row positions	9	68	45
	62*(24+14)	ordered conn_rows	8	76	75
PLA 5	85*(27+10)	none	14	54	30
	85*(27+10)	row positions	10	67	58
	85*(27+10)	conn_row positions	9	72	87
	85*(27+10)	ordered conn_rows	6	70	59
PLA 6	75*(35+29)	none	17	53	50
	75*(35+29)	row positions	19	62	119
	75*(35+29)	conn_row positions	18	64	199
	75*(35+29)	ordered conn_rows	10	73	202
PLA 7	53*(35+29)	none	10	49	26
	53*(35+29)	row positions	13	67	65
	53*(35+29)	conn_row positions	17	58	110
	53*(35+29)	ordered conn_rows	10	80	147
PLA 8	223*(47+62)	none	15	38	1262
	223*(47+62)	row positions	39	55	3933
	223*(47+62)	conn_row positions	39	57	4722
	223*(47+62)	ordered conn_rows	33	60	4769

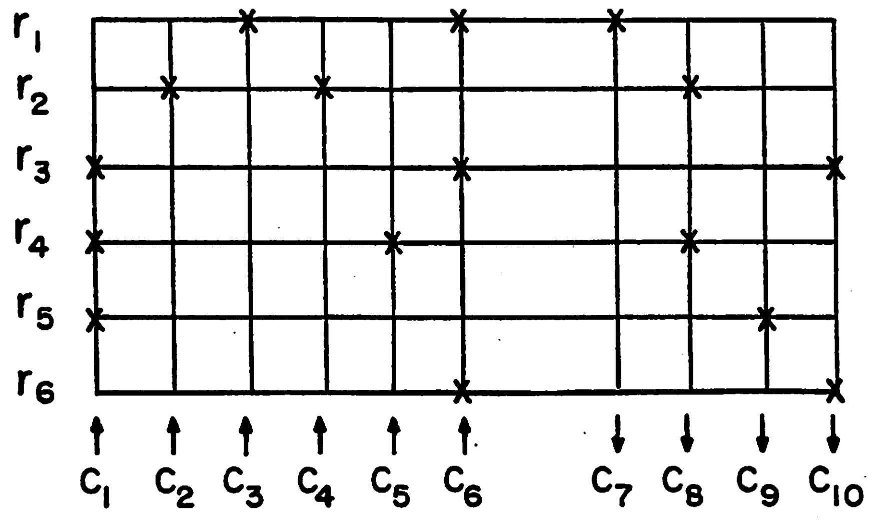
FIGURE CAPTIONS

[5.7b] Edge set $A_6 \cup A_7$.

[5.7c] Edge set $A_5 \cup A_8$.

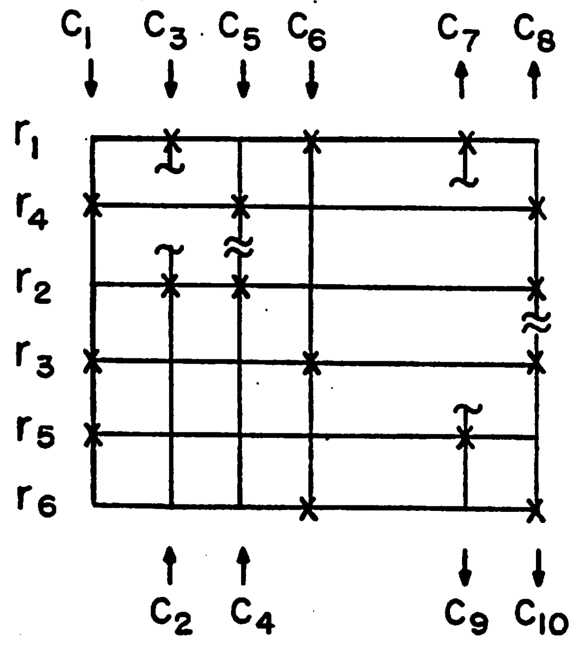
[6.1a] Example of PLEASURE output file for the PLA of fig. 2.1 folded with different constraints: i) no constraints
ii) constrained row positions: $L(r_1) = 1; U(r_1) = 1; L(r_2) = 1; U(r_2) = 3;$ $L(r_3) = 1; U(r_3) = 3;$ $L(r_4) = 4; U(r_4) = 6; L(r_5) = 4; U(r_5) = 6;$ $L(r_6) = 6; U(r_6) = 6.$

[6.1b] Example of PLEASURE output file for the PLA of fig. 2.1 folded with different constraints: i) constrained connection-row positions:
$L_C(c_1) = 1; U_C(c_1) = 6; L_C(c_6) = 6; U_C(c_6) = 6;$
$L_C(c_7) = 1; U_C(c_7) = 1; L_C(c_8) = 1; U_C(c_8) = 3;$
$L_C(c_9) = 4; U_C(c_9) = 6;$ $L_C(c_{10}) = 6; U_C(c_{10}) = 6;$
ii) ordered connection-row positions: $S(c_j) = j$



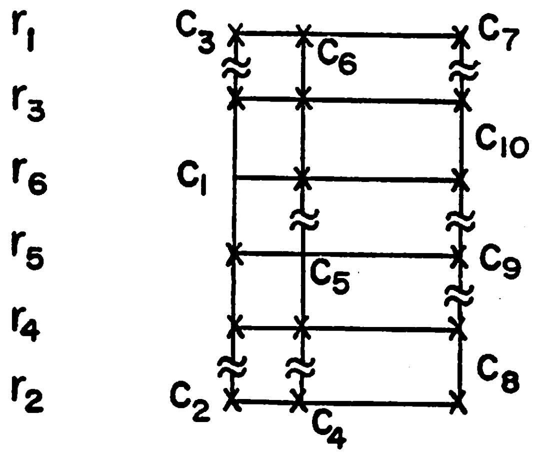
Symbolic representation of a Programmable Logic Array

Fig. 2.1



Simple Folded Array.

Fig. 2.2



Multiple Folded Array

Fig. 2.3

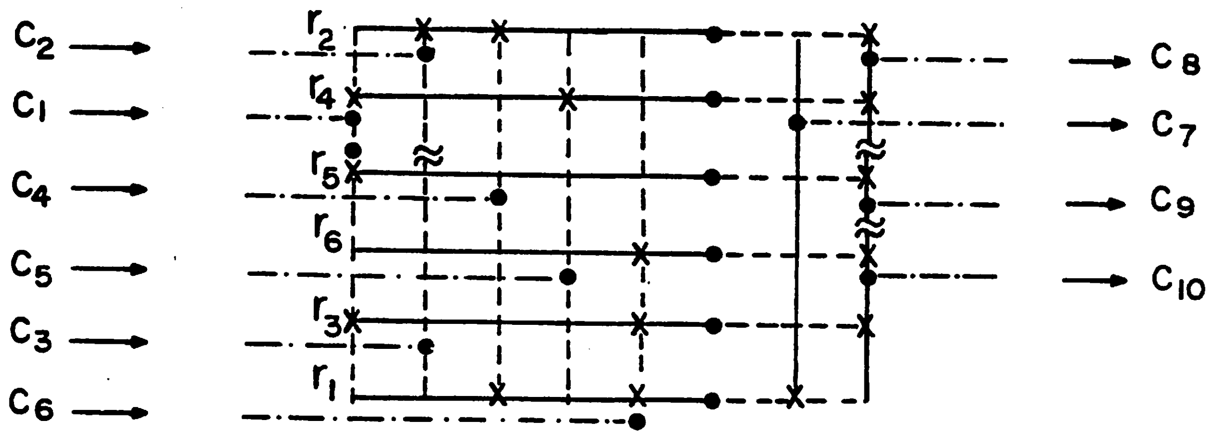


Fig. 2.4

Folded PLA mixed diagram.

LEGEND:	——	metal 1
	---	poly
	-·-·-	metal 2
	x	active device
	≈	cut
	●	contact

(Diffusion ground lines not shown)

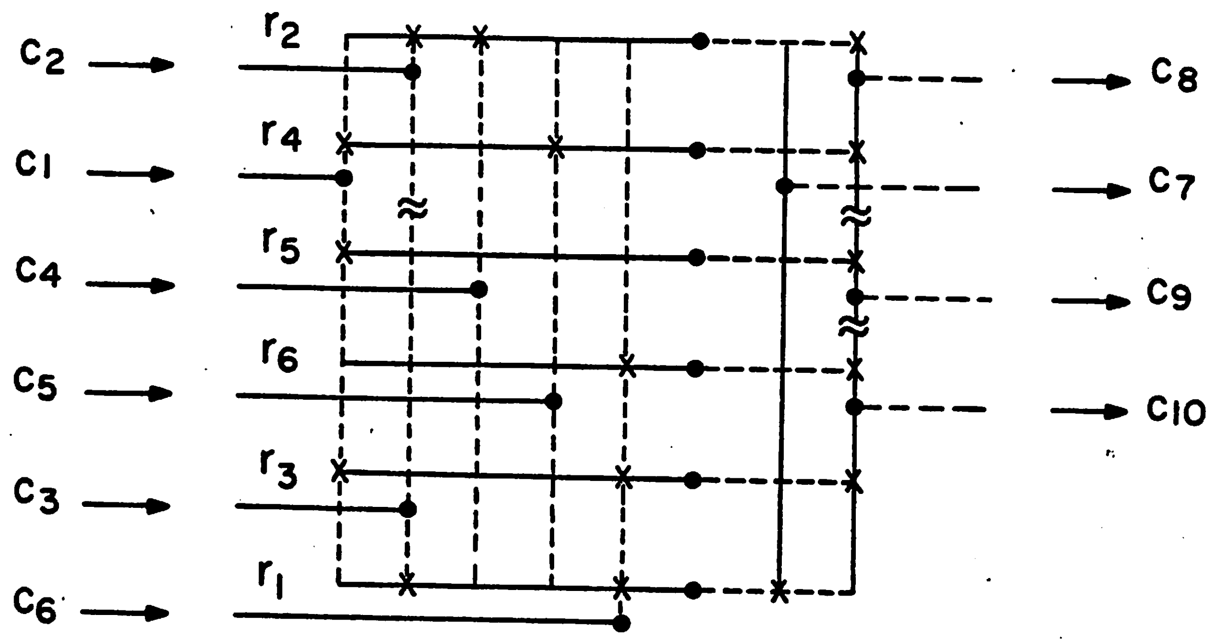
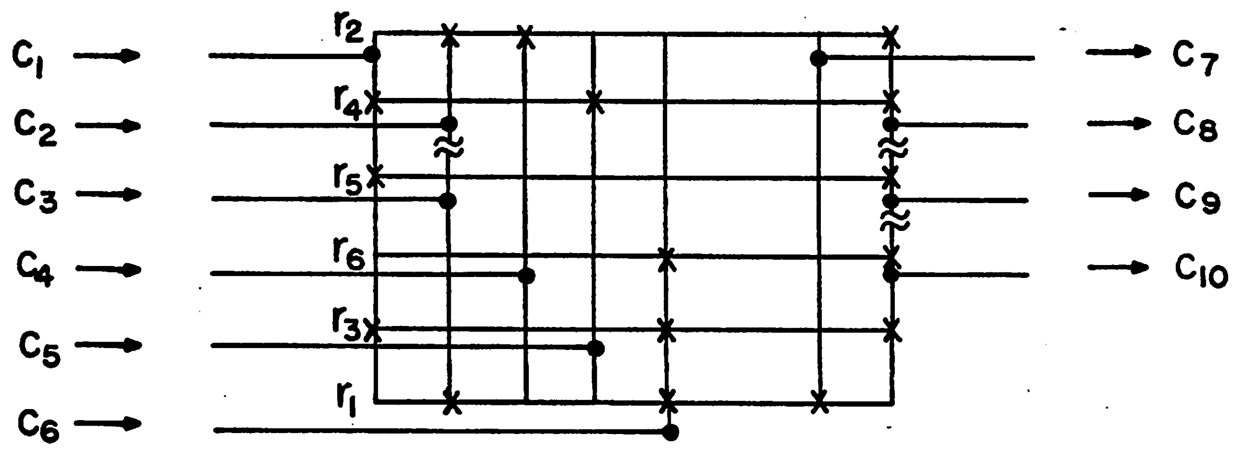


Fig. 2.5 Folded PLA mixed diagram

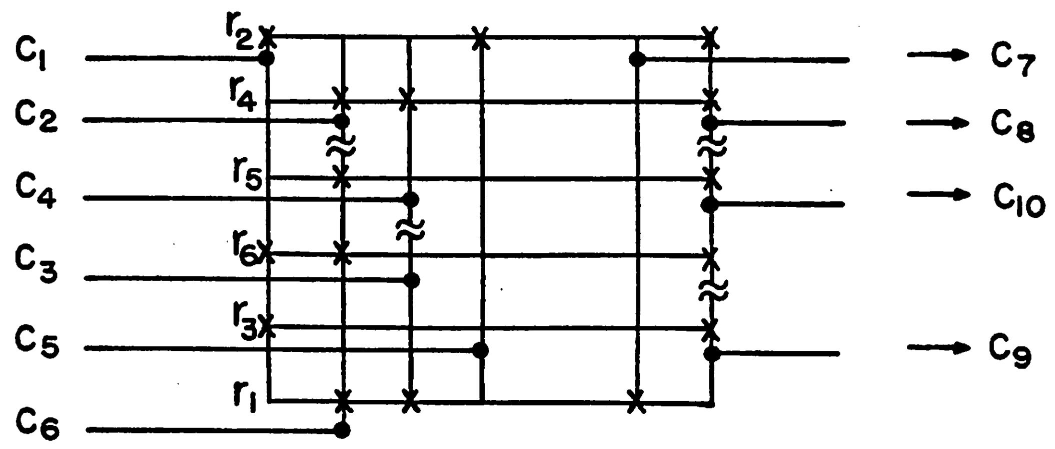
LEGEND:		
	—	metal
	---	poly
	x	active device
	≈	cut
	●	contact

(Diffusion ground lines not shown)



Multiple Folded Array with ordered connection-row assignment

Fig. 2.6



Multiple folded array with bounded connection row assignment.

Connection Row	Lower Bound	Upper Bound
1	1	3
2	1	3
5	4	6
6	4	6
7	1	1
9	4	6

Fig. 2.7

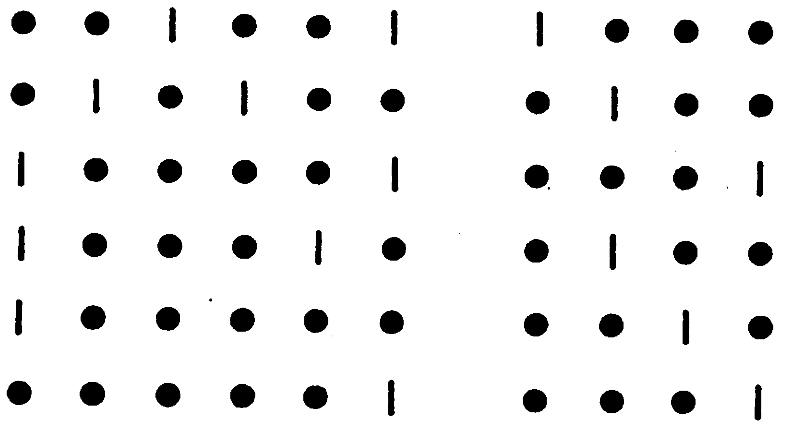


Fig. 3.1-Personality matrix

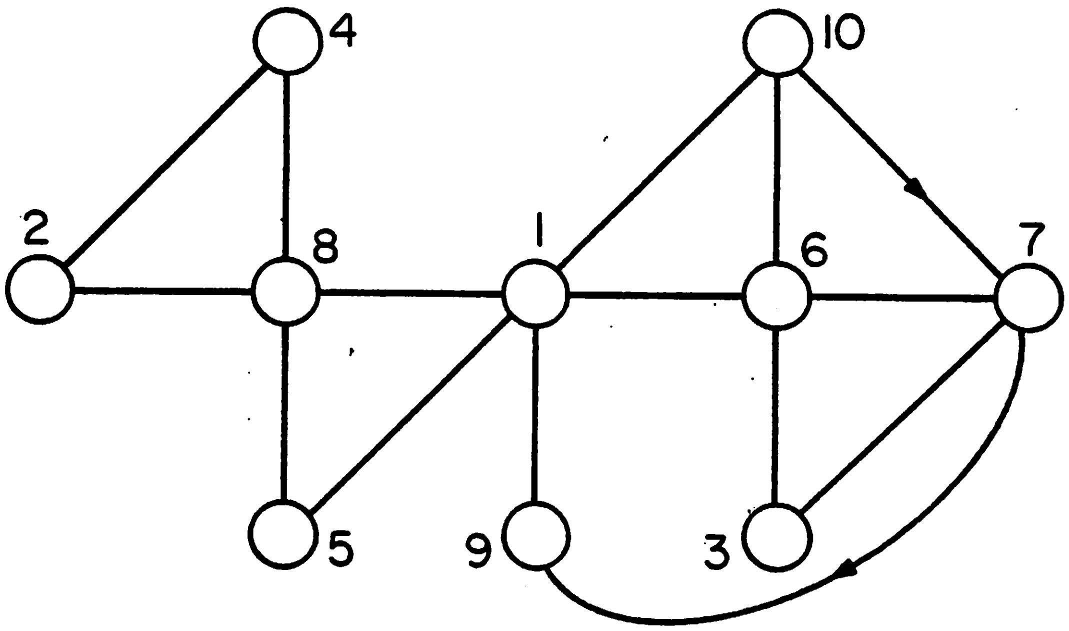


Fig. 3.2 Mixed graph G (V, E, A (O))

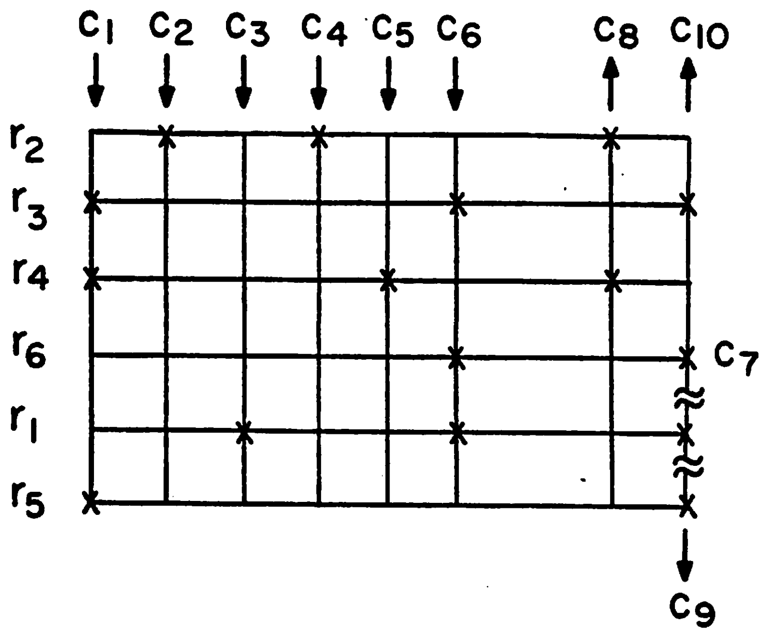


Fig. 3.3 Partially folded array

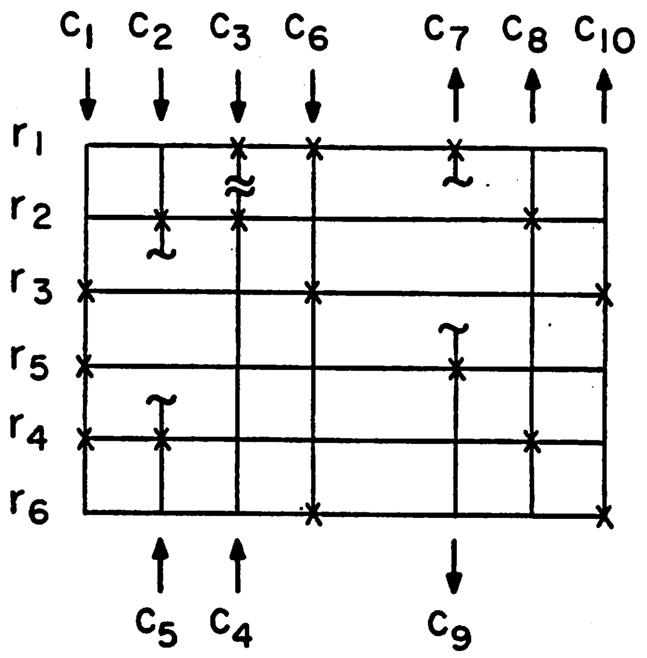


Fig.5.1-Folded logic array with bounded row assignment

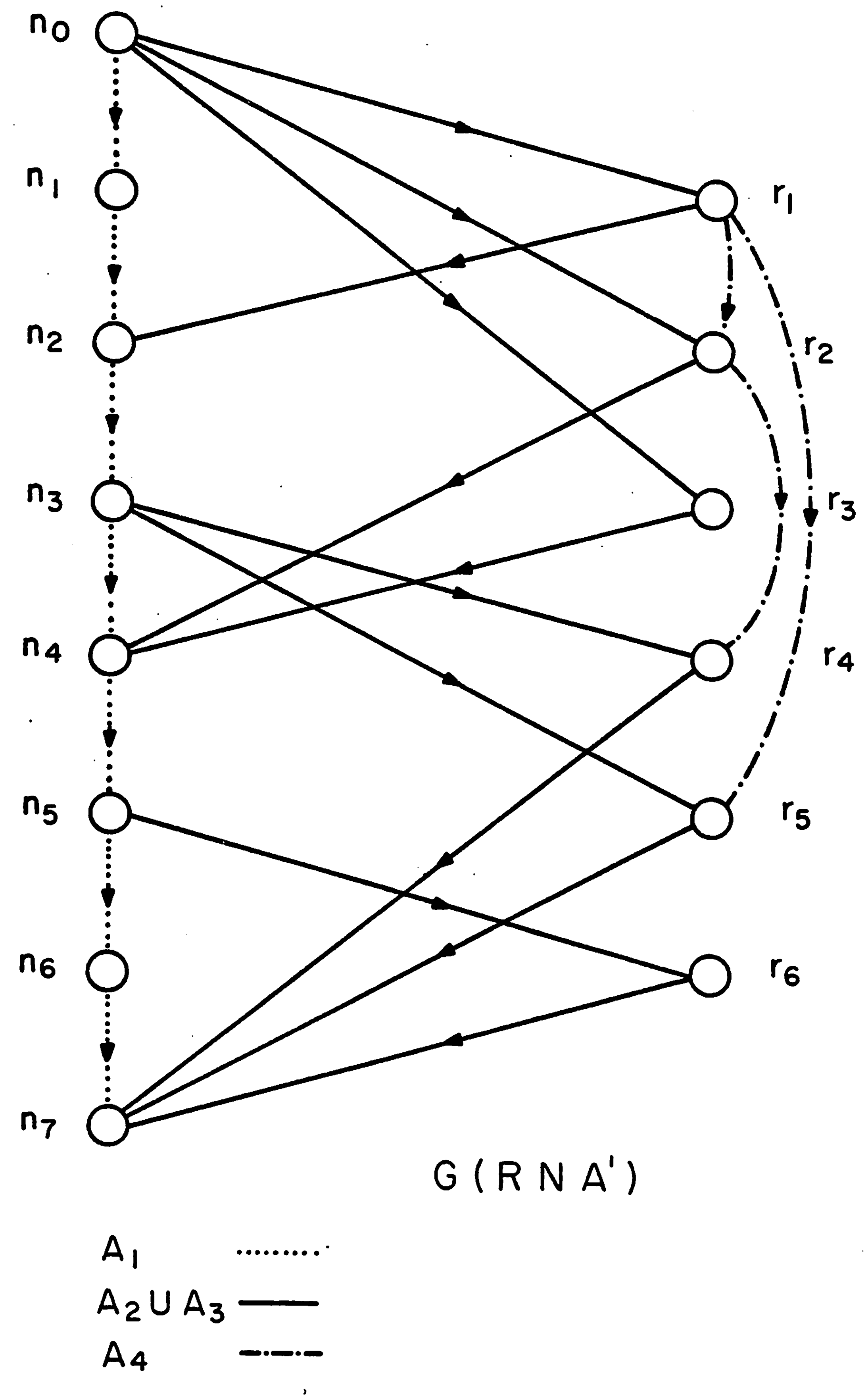


Fig. 5.2a

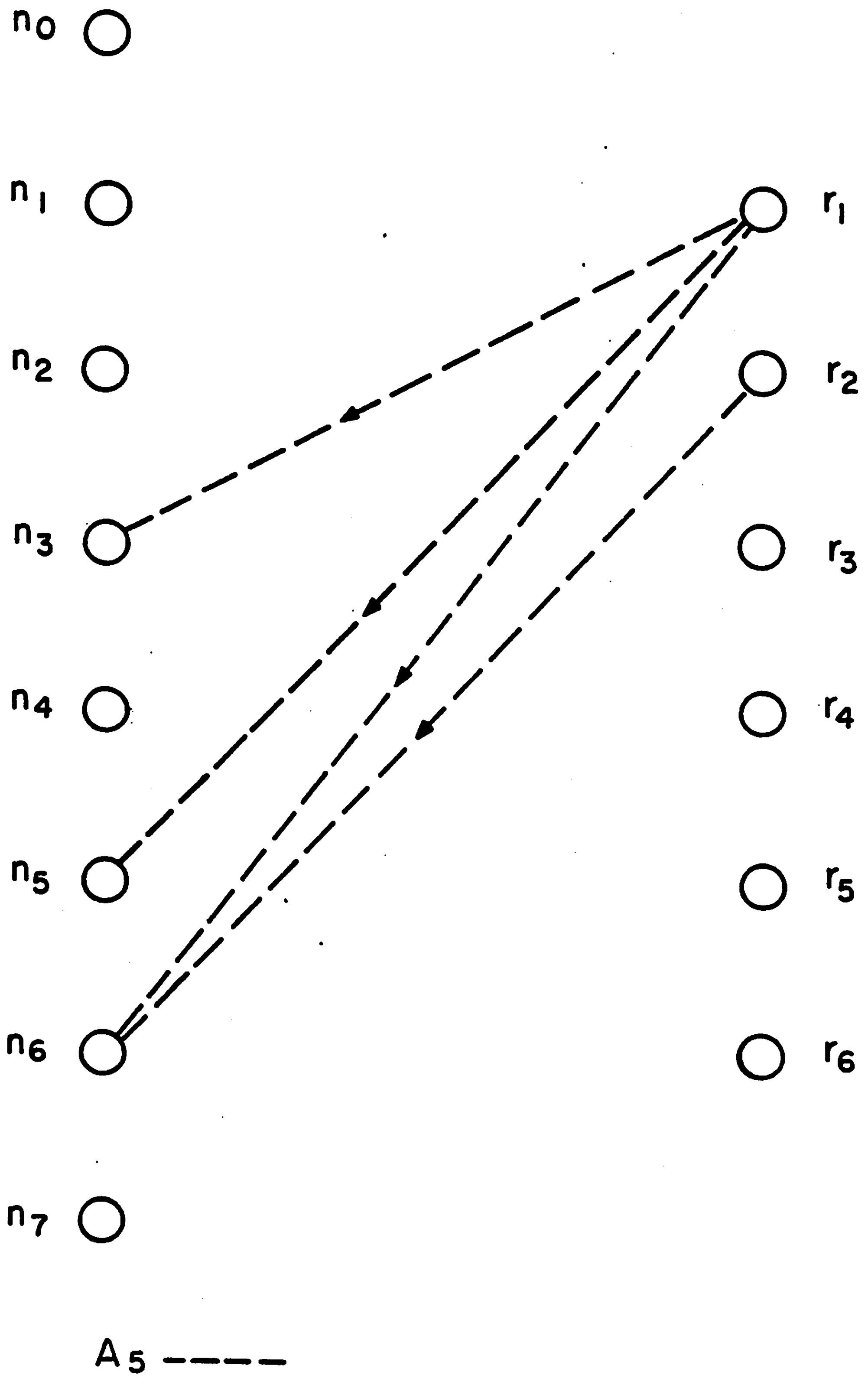


Fig. 5.2b

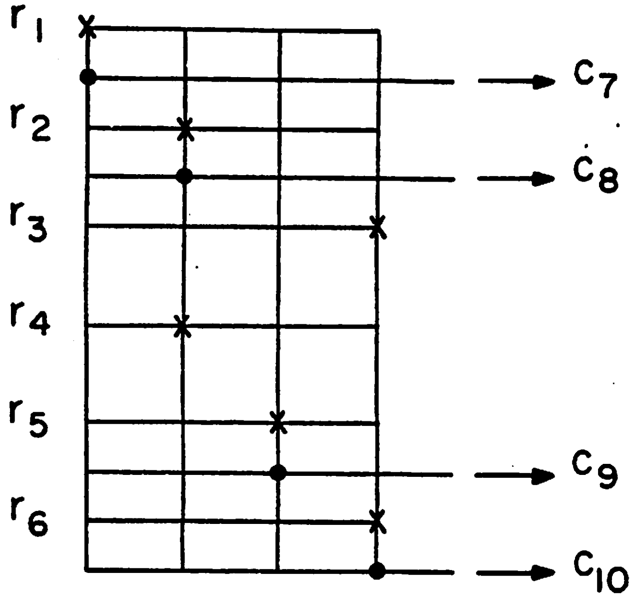


Fig.5.3-Unfolded OR array

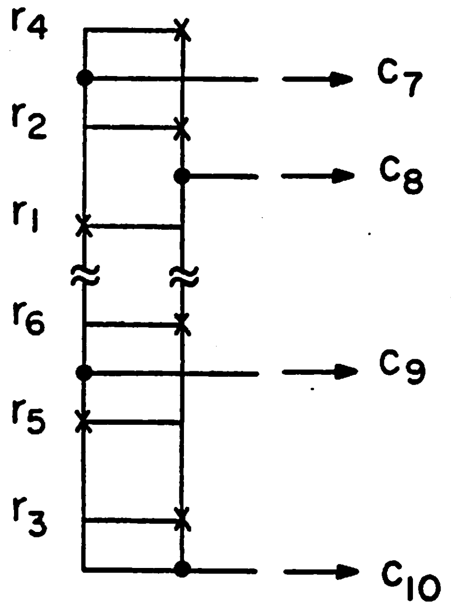
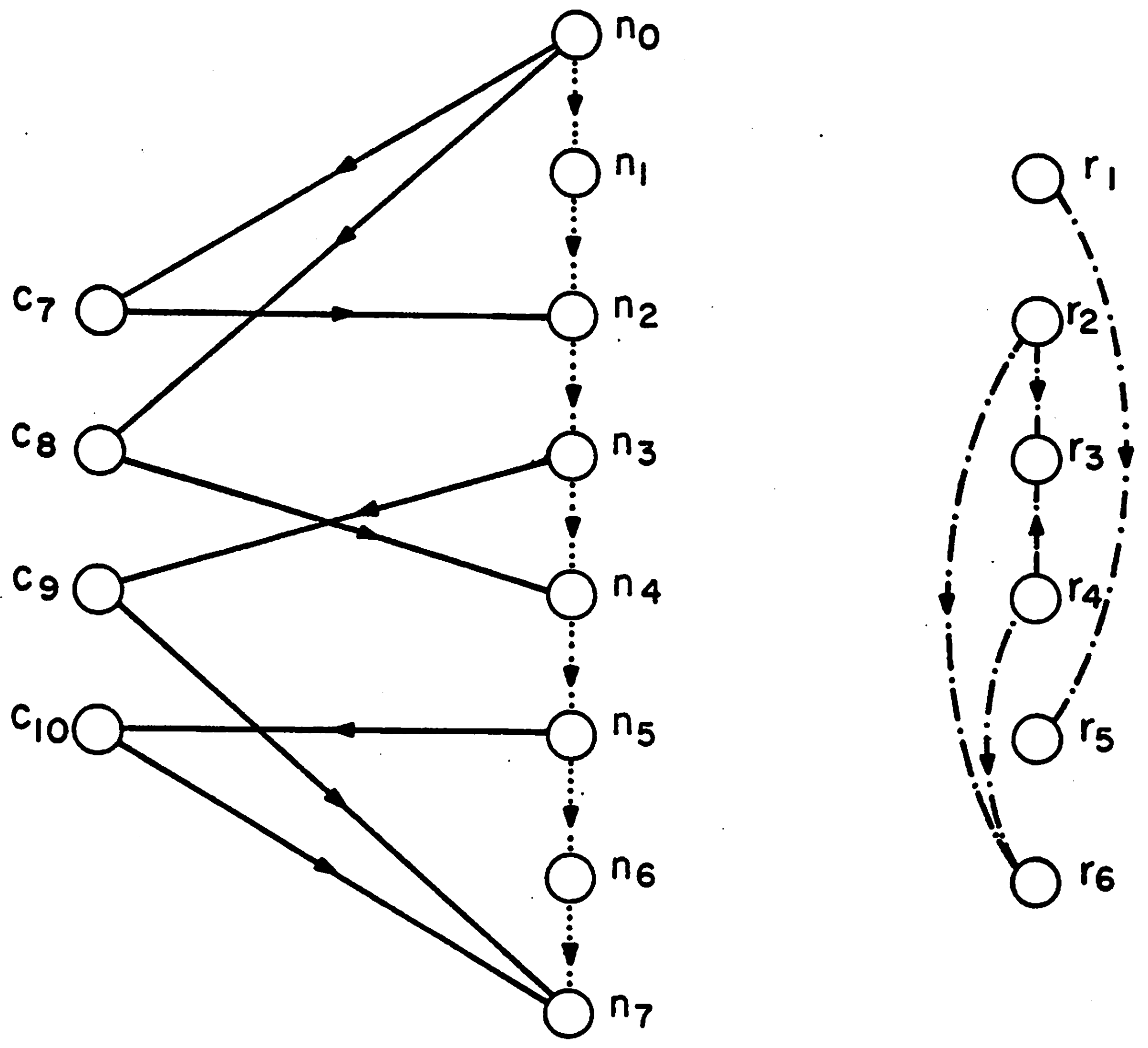


Fig. 5.4- Folded OR array with bounded connection row assigment



A_1	....
$A_2 \cup A_3$	——
A_4	-·-·-

Fig. 5.5 a

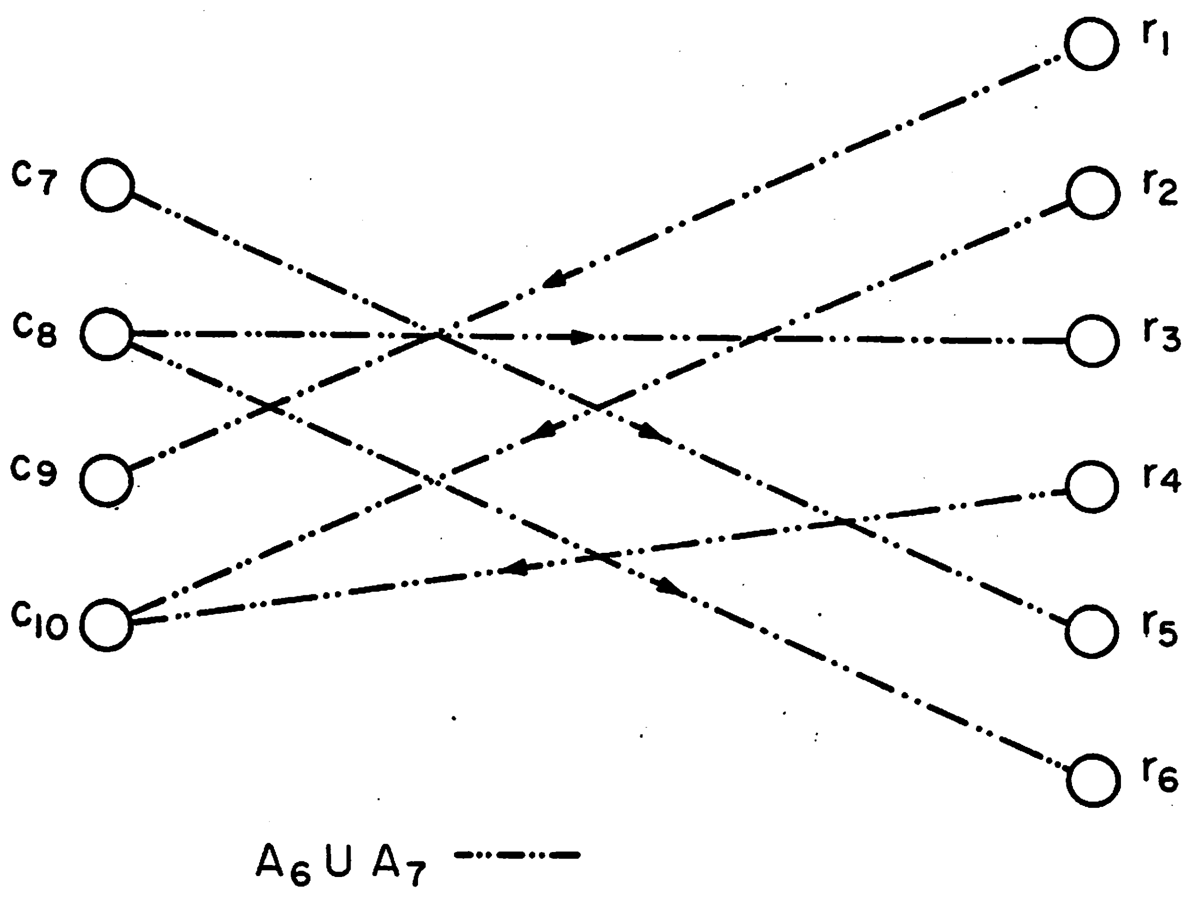


Fig. 5.5 b

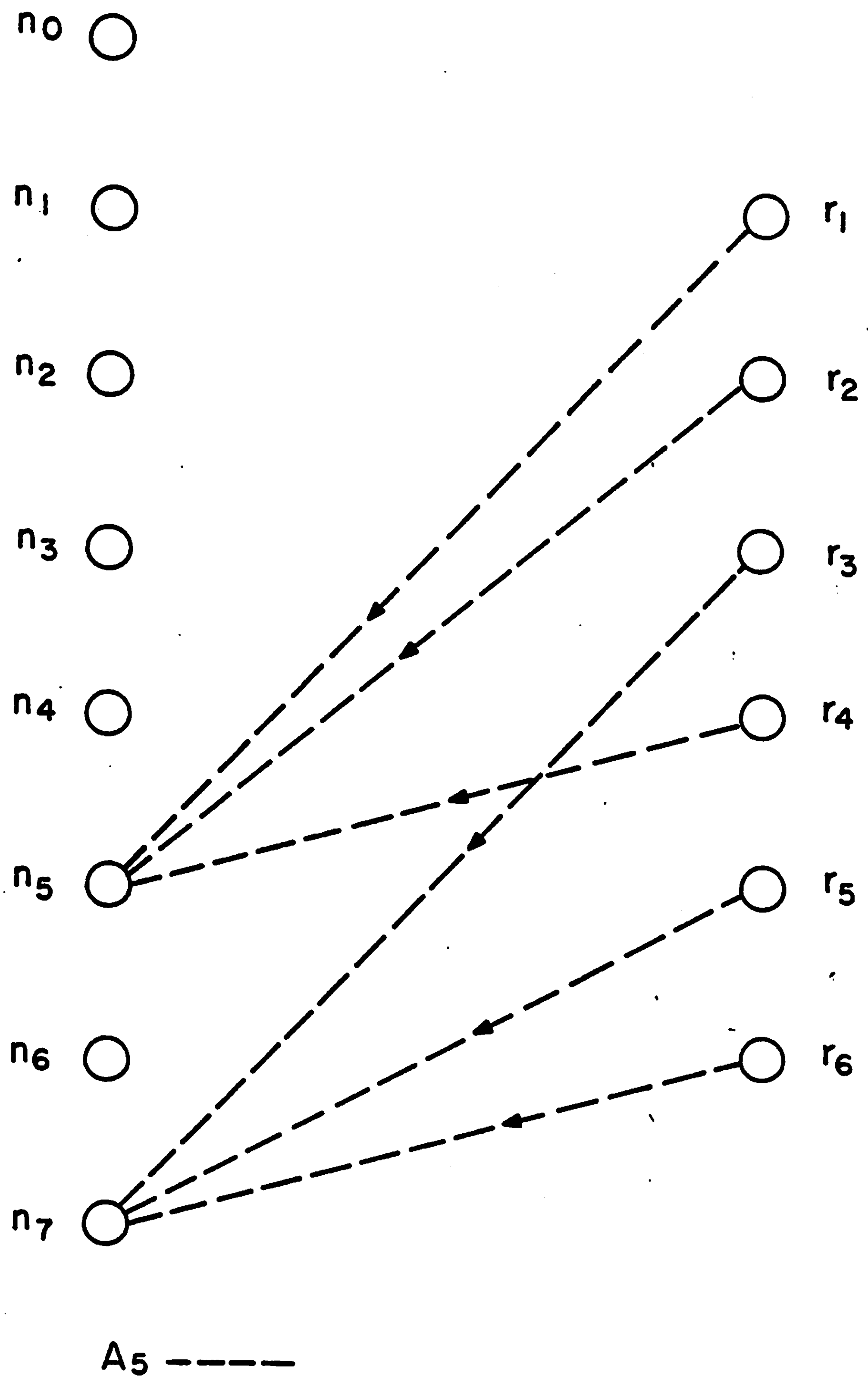


Fig. 5.5c

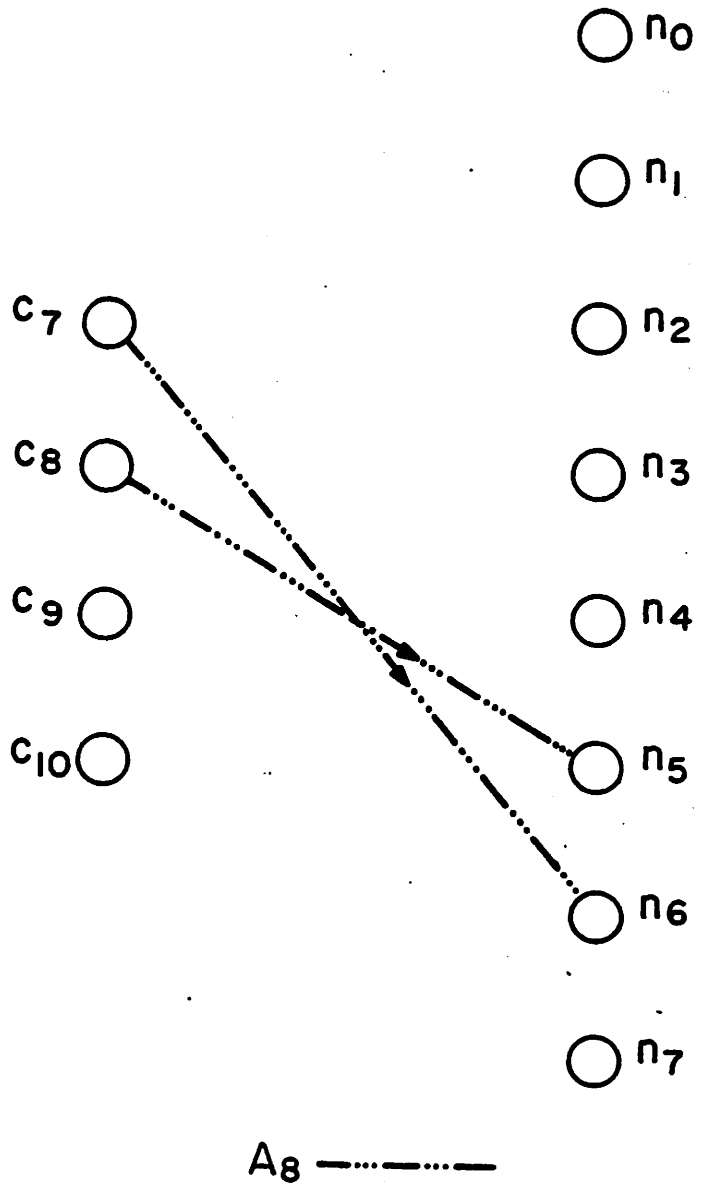


Fig. 5.5d

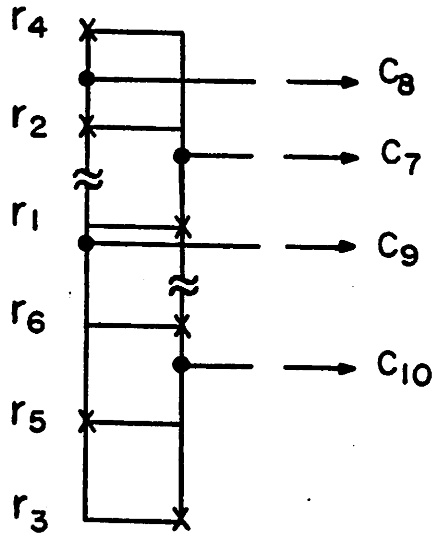


Fig. 5.6- Folded OR plane implementation.

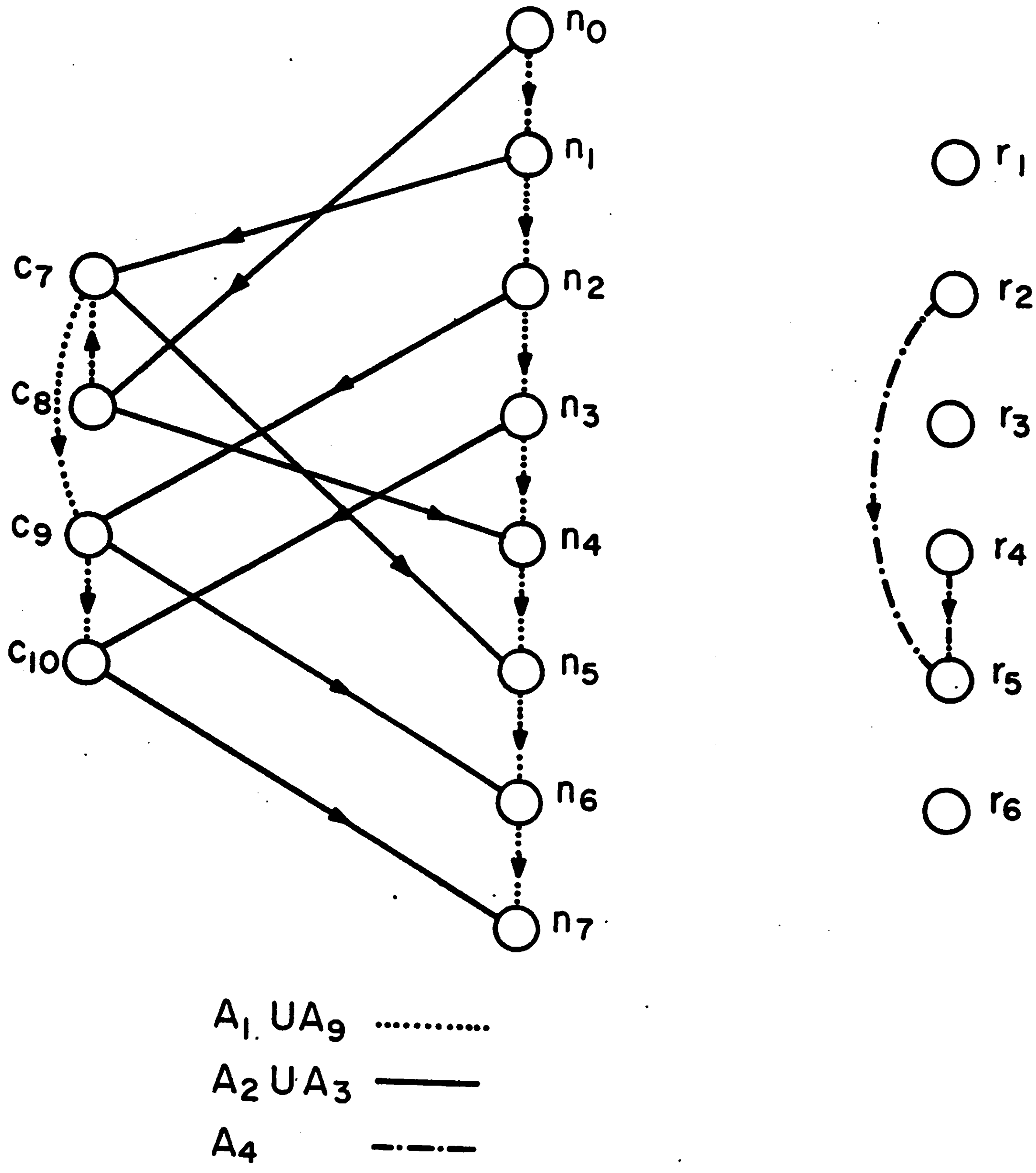

n_0
n_1
n_2
n_3
n_4
n_5
n_6
n_7
c_7
c_8
c_9
c_{10}
r_1
r_2
r_3
r_4
r_5
r_6
$A_1 \cup A_9$
$A_2 \cup A_3$
A_4

Fig. 5.7a

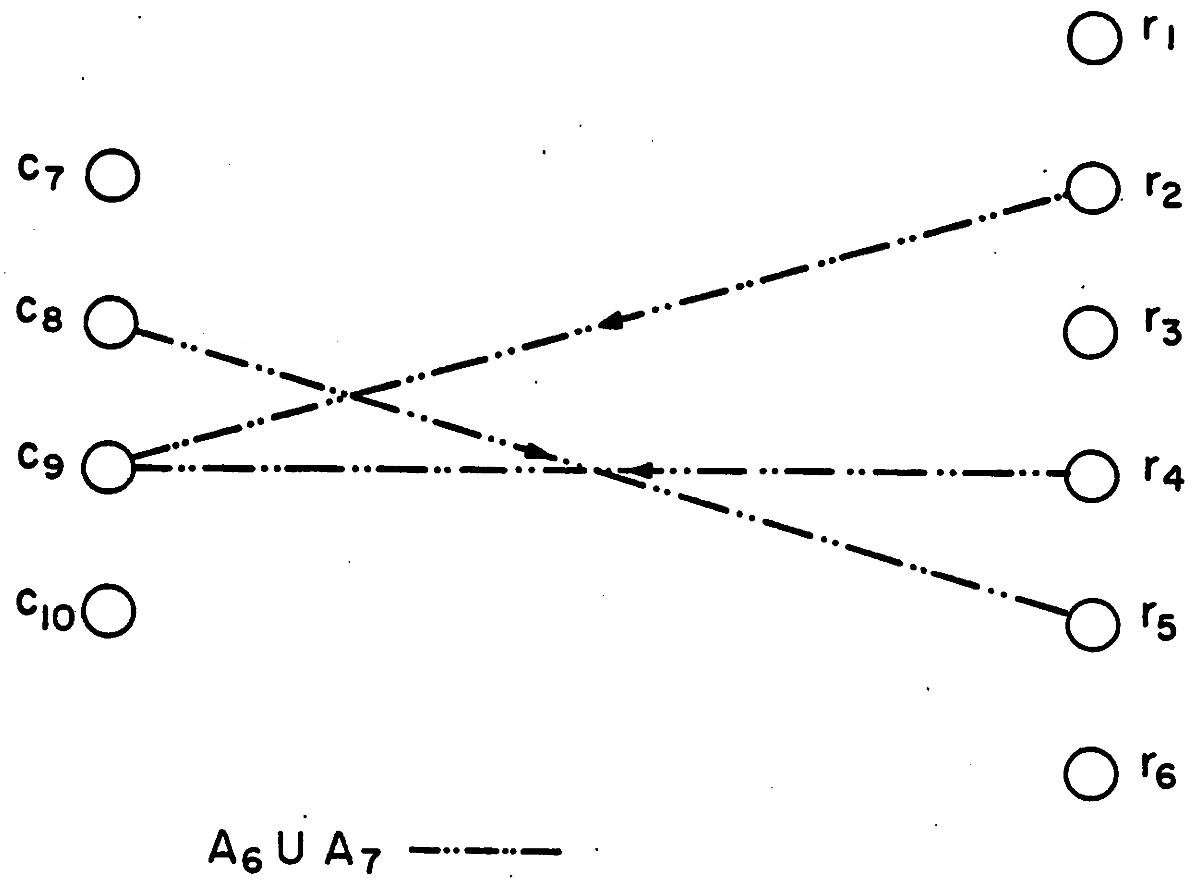


Fig. 5.7b

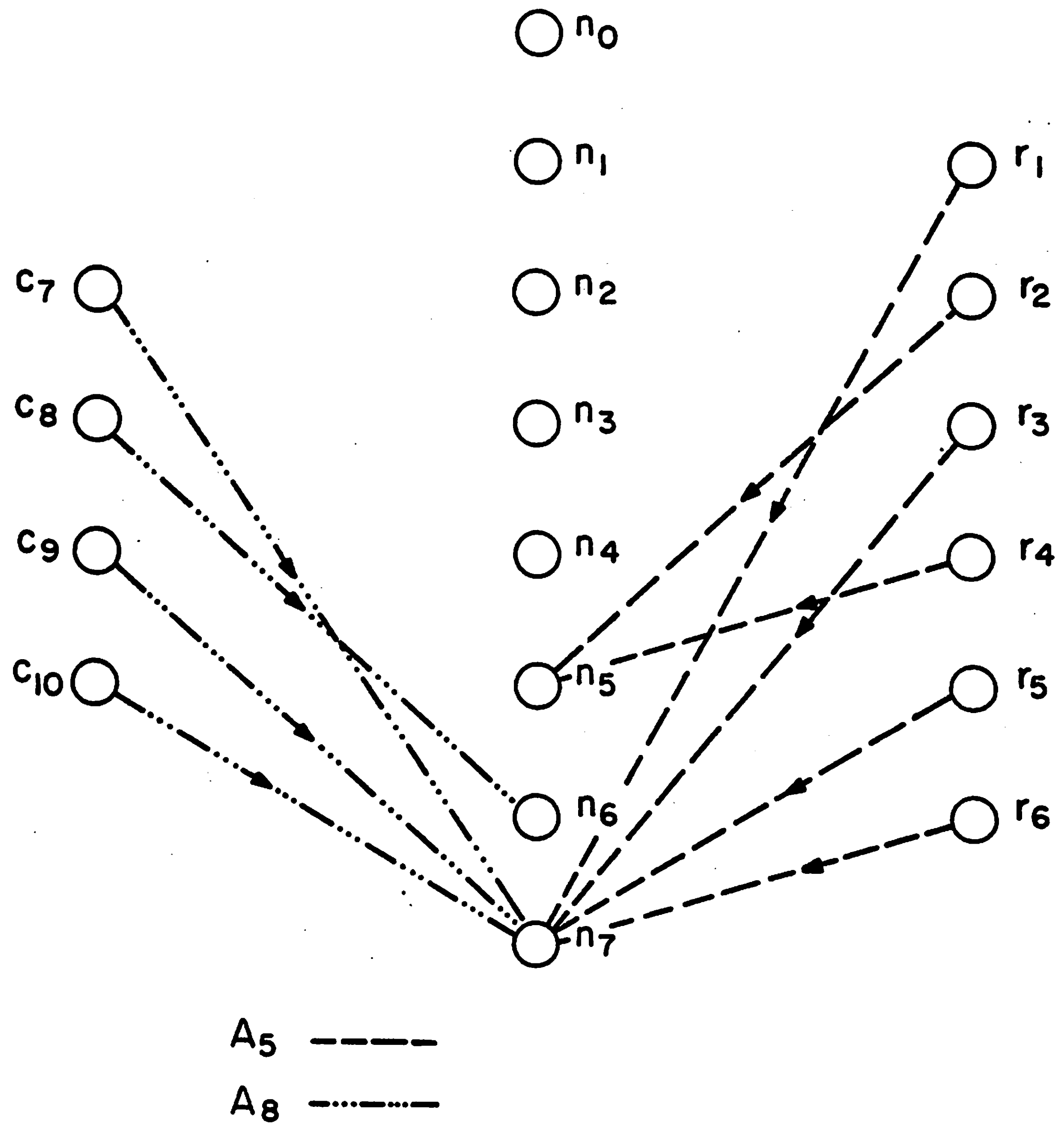


Fig. 5.7 c

```
******************************************************************
*                                                                *
*      PLEASURE : PLA TOPOLOGICAL COMPACTION ENVIRONMENT        *
*                                                                *
*            UNIVERSITY OF CALIFORNIA , BERKELEY                 *
*                                                                *
******************************************************************

         RELEASE FOR  : University of California, Berkeley
         UPDATE   #   :   27
         RELEASE DATE : Mon Nov 22 19:42:01 1982

FOLDING REQUESTED:

        MULTIPLE COLUMN FOLDING IN THE AND PLANE

        MULTIPLE COLUMN FOLDING IN THE OR PLANE


COLUMN FOLDINGS:


       ORDERED COLUMN FOLDING LIST #   1

             7
            10
             9
             8

       ORDERED COLUMN FOLDING LIST #   2

             3
             1
             2

       ORDERED COLUMN FOLDING LIST #   3

             6
             5
             4

COLUMNS FROM THE TOP            ROWS FROM THE LEFT

       3                                1
       6                                3
       7                                6
                                        5
                                        4
                                        2

  PERSONALITY MATRIX


       !0  1
       10  I
       -!  1
       0-  1
       !!  I
       10  I

Folded PLA takes 30% of the original area
```

```
******************************************************************
*                                                                *
*      PLEASURE : PLA TOPOLOGICAL COMPACTION ENVIRONMENT        *
*                                                                *
*            UNIVERSITY OF CALIFORNIA , BERKELEY                 *
*                                                                *
******************************************************************

         RELEASE FOR  : University of California, Berkeley
         UPDATE   #   :   27
         RELEASE DATE : Mon Nov 22 19:42:01 1982


FOLDING REQUESTED:

        MULTIPLE COLUMN.FOLDING IN THE AND PLANE

        MULTIPLE COLUMN FOLDING IN THE OR PLANE

        COLUMN FOLDING WITH CONSTRAINED ROW POSITIONS


COLUMN FOLDINGS:


       ORDERED COLUMN FOLDING LIST #   1

             7
             8
             9

       ORDERED COLUMN FOLDING LIST #   2

             3
             2
             5

       ORDERED COLUMN FOLDING LIST #   3

             4
             1


COLUMNS FROM THE TOP                    ROWS FROM THE LEFT

       3                                        1
       4                                        2
       6                                        3
       7                                        4
      10                                        5
                                                6

  PERSONALITY MATRIX


       !-0  1~
       !0-  I~
       -10  ~I
       11-  1~
       -0-  I~
       --1  ~I

Folded PLA takes 50% of the original area
```

Fig. 6.1a

```
*************************************************************************
*                                                                       *
*        PLEASURE : PLA TOPOLOGICAL COMPACTION ENVIRONMENT              *
*                                                                       *
*               UNIVERSITY OF CALIFORNIA , BERKELEY                     *
*                                                                       *
*************************************************************************

FOLDING REQUESTED:

        MULTIPLE COLUMN FOLDING IN THE AND PLANE

        MULTIPLE COLUMN FOLDING IN THE OR PLANE

        COLUMN FOLDING WITH CONSTRAINED CONTACT POSITIONS


COLUMN FOLDINGS:


       ORDERED COLUMN FOLDING LIST #   1

              8
              9

       ORDERED COLUMN FOLDING LIST #   2

              7
             10

       ORDERED COLUMN FOLDING LIST #   3

              2
              5

       ORDERED COLUMN FOLDING LIST #   4

              4
              3
              1


COLUMNS FROM THE TOP                    ROWS FROM THE LEFT

              2                                 2
              4                                 1
              6                                 6
              7                                 4
              8                                 5
                                                3


CONTACTS ON THE LEFT PLANE              CONTACTS ON THE RIGHT PLANE

             4                                     7
             2                                     8
             3
             1
             5                                     9
             6                                    10


    PERSONALITY MATRIX


       !o-  ~I
       -!0  i~
       --1  I~
       11-  ~i
       -0-  ~I
       -10  I~

Folded PLA takes [illegible] of the original area
```

Fig. 6.1b

```
*************************************************************************
*                                                                       *
*        PLEASURE : PLA TOPOLOGICAL COMPACTION ENVIRONMENT              *
*                                                                       *
*               UNIVERSITY OF CALIFORNIA , BERKELEY                     *
*                                                                       *
*************************************************************************

FOLDING REQUESTED:

        MULTIPLE COLUMN FOLDING IN THE AND PLANE

        MULTIPLE COLUMN FOLDING IN THE OR PLANE

        ORDERED FOLDING IN THE LEFT ARRAY

        ORDERED FOLDING IN THE RIGHT ARRAY


COLUMN FOLDINGS:


       ORDERED COLUMN FOLDING LIST #   1

              7
              9

       ORDERED COLUMN FOLDING LIST #   2

              3
              4
              5


COLUMNS FROM THE TOP                    ROWS FROM THE LEFT

              1                                 1
              2                                 6
              3                                 5
              6                                 2
              7                                 4
              8                                 3
             10

CONTACTS ON THE LEFT PLANE              CONTACTS ON THE RIGHT PLANE

             1                                     7
             2                                     8
             3                                     9
             4                                    10
             5
             6

   PERSONALITY MATRIX


       --!0  1~~
       ---1  ~~I
       0---  I~~
       -1o-  ~I~
       1-1-  ~I~
       1--0  ~~I

Folded PLA takes 70% of the original area
```