MULTIPLE FOLDING OF PROGRAMMABLE LOGIC ARRAYS

Giovanni De Micheli and Alberto Sangiovanni-Vincentelli
Department of EECS
University of California at Berkeley
Berkeley CA 94720

Abstract: This paper addresses the problem of optimizing the silicon area and the performance of large Programmable Logic Arrays. In particular we describe a general method for compacting a logic array called **multiple row and column folding**, and two physical implementations for multiply folded arrays. We present a graph theoretic representation of the multiple folding problem, that is used to develop a heuristic strategy for a multiple folding algorithm.

1. INTRODUCTION

Programmable Logic Arrays are extensively used in the structured design of Very Large Scale Circuits and Systems [1] [2]. Though heuristic minimizers [3] [4] allow to express large switching functions as minimal sets of logical implicants, their physical implementation may still be too expensive in terms of silicon area. Large logic arrays are in general very sparse: the number of "cares" is much smaller than the number of "don't cares"[5]. A straightforward physical implementation results in a significant waste of the silicon area not directly contributing to the implementation of the logic function. The wasted area reduces circuit yield and degrades the time performance of the PLA by introducing unnecessary parasitics.

In this paper we address the problem of optimizing the area used by a PLA, by means of row and column folding [5] [6]. Wood presented for the first time a folded PLA implementation in [5], and Hachtel et al. an algorithm for PLA folding in [6]. 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].

We presented in [12] an algorithm for multiple constrained folding: columns and/or rows are folded subject to a set of constraints on their position. Constrained multiple folding allows to compact the PLA area while ensuring an easy routing of the folded array and therefore represents an effective tool for VLSI design.

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 [13] 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] [7]. 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. 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] [14] 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 [12].

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.

We defined in [12] a multiple constrained column folding problem related to the ordering of the connection-rows. In particular folding is constrained so that connection-rows can be positioned according to a given sequence or satisfying given position bounds.

Multiple row folded PLAs can be implemented with a single-poly, single-metal technology [13]. 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

CH1845-7/83/0000-1026 $1.00

increased complexity of input and output routing. On the other hand 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.

Multiply row and column-folded arrays can be implemented with the described architectures, provided that only columns in the external planes are multiply folded.

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 $\{c_i, \ i = 1, 2, \cdots, nc\}$ $(\{r_i, \ i = 1, 2, \cdots, nr\})$ 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. :
$U = \{c \mid \exists k \ s.t. \ c \in o_k\}$ $(U = \{r \mid \exists k \ s.t. \ r \in o_k\})$. 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 $QR(O)$ $(QC(O))$ [16]. 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

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:

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

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 [12].

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. ■

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 [12]. 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 [17]. We therefore propose a heuristic algorithm that can be considered an extension of the simple folding algorithm presented in [6].

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:

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

A detailed description of the algorithm for simple column folding is given in [6]. 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 $D(v_2)$. 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:

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

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 $\tilde{D}(v_2)$. Therefore, an upper bound on the number of ancestor-descendant relations induced by the column fold is:

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

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 $\tilde{D}(v_{i,k})$ and the ancestors $A(v_{i,k-1})$ become ancestors of the vertices in $\tilde{D}(\bar{v})$. Therefore, an upper bound on the number of ancestor-descendant relations is:

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

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:

$$\hat{v} = \underset{v \in \bar{V}}{arg\,\min} \left(|\tilde{D}(v)| + |A(v)| \right)$$

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 Folding Algorithm and the candidate selection strategy applies "mutatis mutandis" to the multiple unconstrained row folding problem. ■

The Folding 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 descen-

dant 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 :

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

where $\alpha = \frac{C(O)-1}{C(O)}$ and $\beta = \frac{R(O)-1}{R(O)}$ 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 $C(O)$ $(R(O))$ 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. EXPERIMENTAL AND CONCLUDING REMARKS

Computer program PLEASURE implements the algorithms for multiple and/or constrained folding. The output of the program is a symbolic array, showing the split rows and columns, the positions of the personalizing transistors and the contacts between columns and connection-rows according to the required PLA architecture. PLEASURE is interactive: the user has the option to ask for column folding only, row folding only or row and column folding in a sequence. Different requirements on folding (e.g.: simple/multiple) can be set in the AND plane and in the OR plane independently. Simple row folding supports predefined structures, such as AND-OR-AND or OR-AND-OR architectures [12].

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.

Program PLEASURE has been successfully tested on a large set of industrial PLAs. Results are reported in [12].

6. ACKNOWLEDGEMENTS

This research has been sponsored by Fairchild and GTE corporations, NSF under subcontract #392741C-1 and DARPA under contract #25976.

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

[6] 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

[7] 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

[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] G.De Micheli and A.L.Sangiovanni Vincentelli "PLEASURE: A Computer Program for Simple/Multiple Constrained/Unconstrained Folding of Programmable Logic Arrays" *Memorandum* UCB/ERL No. M82/57

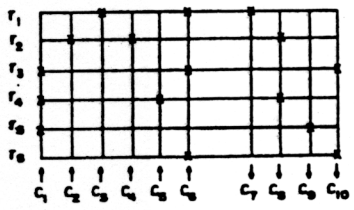
[13] C.Mead and L.Conway "*Introduction to VLSI Systems*" Addison Wesley 1980

[14] G. De Micheli " Pleasure: A Program for topological compaction of PLAs" *Internal Report*, Harris Corporation , 1980

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

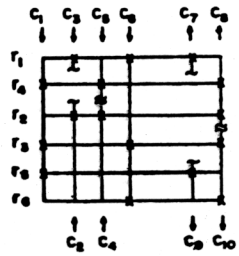
[16] A.V.Aho J.E.Hopcroft and J.D.Ullman "*The Design and Analysis of Computer Algorithms*" Addison Wesley 1974

[17] 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



Symbolic representation of a Programmable Logic Array

Fig. 2.1



Simple Folded Array.

Fig. 2.2

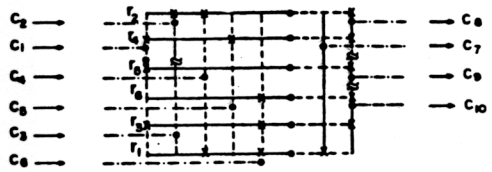


Fig. 2.4
Folded PLA mixed diagram.

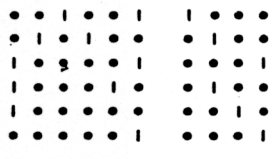


Fig. 3.1- Personality matrix

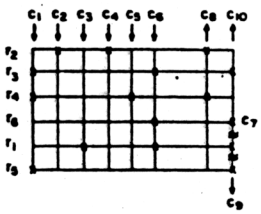


Fig. 3.3 Partially folded array

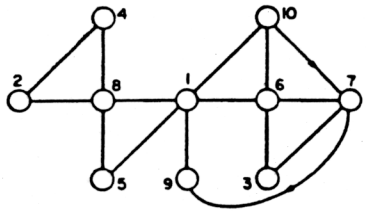


Fig. 3.2 Mixed graph G (V, E, A (O))