GRAPH THEORETIC ALGORITHM FOR VLSI REGULAR STRUCTURES FOLDING

L. Cheremisinova

Institute of Engineering Cybernetics of National Academy of Sciences of Belarus, Surganov str., 6, Minsk, 220012, BELARUS, cld@newman.bas-net.by

ABSTRACT

The problem under consideration is to reduce the area of the layout of regular VLSI structures by means of their folding. Some results of cutting down exhaustive computational efforts on the search for the optimal regular structures folding have been described. An efficient graph theoretic algorithm for one-dimensional unconstrained simple folding is presented.

1. INTRODUCTION

Structured logic refers to logic forms that exhibit a high degree of regularity in their layout and interconnections. The most widely used regular structures are Programmable Logic Array (PLA), Gate Matrix, Weinberger Matrix. All these forms have a two-dimensional structure consisting of an array of rows and columns. There are transistors in intersections of some rows and columns.

The use of such regular structures makes it possible to automatically generate the layout from its functional description. The price paid for the structural regularity is a larger chip area because the layouts obtained are sparse: a large percentage of the row-column intersections are not personalized. Several techniques have been proposed for reducing the area required.

The proposed paper deals with the problem of optimizing the area of the layout by means of its folding [1-5]. The folding problem is proved to be NP-complete. The classification of a problem as NP-complete is not sufficient reason to develop only heuristic optimizing methods. In many cases it is interesting to get just an optimal solution.

Some results of reduction of exhaustive computational efforts on the search for the optimal PLA folding have been described. The problem of the column (or row) folding [1-4] is considered. In particular we shall focus here on one of the kinds of regular structures folding which is referred to as a simple folding [1, 3, 4, 5]. Folding can be done in one or two dimensions. The objective of one-dimensional column (row) folding is to find a permutation of the rows (columns) such that the set of columns (rows) can be placed in the minimum number of vertical (horizontal) lines. In this paper an efficient graph theoretic algorithm for one-dimensional unconstrained simple folding is presented.

2. THE ARRAY FOLDING PROBLEM

In the folding problem we are given a Boolean matrix B having sets $C(B)$ and $R(B)$ of their columns and rows. A 1 in the position i,j of the matrix B means there is appropriate crosspoint (transistor) between row i and column j in the matrix. A subset $R(c_i)\subseteq R(B)$ is associated with each column $c_i \in C(B)$, it represents the set of rows that are connected to column c_i: $r_i \in R(c_i) \leftrightarrow b^i_j$ 1. And a subset $C(r_i)\subseteq C(B)$ is defined similar manner. The objective of matrix folding is to arrange the columns and rows as a matrix in the plane using the least number of columns or the least number of rows or both assigning columns or rows to the same vertical or horizontal lines.

Example 1. PLA is described in symbolic form by a Boolean matrix B which has as many columns as different inputs x_i and outputs y_i and as many rows as different product terms' rows r_i. The column's array $C(B)$ of B is divided into 2 parts X and Y that correspond to input x_i and output y_i variables. The set $C(B)$ of columns is the union set of inputs and outputs, i.e. $C(B)=X\cup Y$. PLA matrix shown in fig. 1 is described by the following Boolean matrix B:

$$B = \begin{array}{c} r1 \\ r2 \\ r3 \\ r4 \\ r5 \\ r6 \end{array} \left[\begin{array}{cccccc|cc} 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right]$$

Two columns c_i and c_j (rows r_i and r_j) are disjoint if $R(c_i)\cap R(c_j)=\varnothing$ ($C(r_i)\cap C(r_j)=\varnothing$). A pair of disjoint columns is defined to be column folding pair c_{k1},c_{k2}. Only such two columns can be folded together. Let us distinguish an unordered folding pair c_{k1},c_{k2} from an ordered one denoted as $<c_{k1},c_{k2}>$. Unordered folding pair c_{k1},c_{k2} implies

one of two appropriate ordered folding pairs: $<c_{k1},c_{k2}>$ or $<c_{k2},c_{k1}>$. The ordering a pair c_{k1},c_{k2} signifies the choose between the last two ordered pairs.

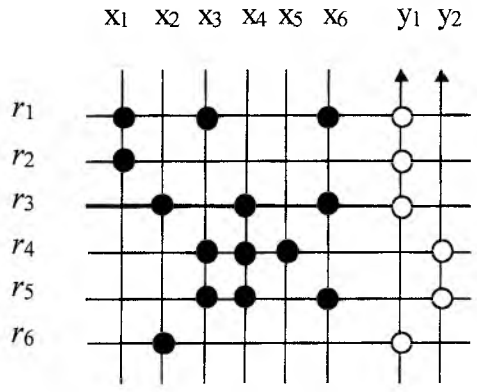

Figure 1. PLA matrix

Each ordered column folding pair $<c_{k1},c_{k2}>$ induces a relation $R(c_{k1})<R(c_{k2})$ on the set of rows $R(B)$. It forces all rows of $R(c_{k1})$ to be on the top of the rows of $R(c_{k2})$ in the folded matrix B. That defines a partial ordering (irreflexive, asymmetric) R_k of rows: $R_k=R(c_{k1})\times R(c_{k2})$.

An ordered folding set $C_o=\{<c_{11},c_{12}>,<c_{21},c_{22}>,\ldots, <c_{p1},c_{p2}>\}$ is a set of disjoint ordered folding pairs (all c_{ij} are different). Ordered folding set C_o induces an ordering relations R_{Co} given by the union of the partial orderings $R(c_{i1}) \times R(c_{i2})$ induced by the pairs of C_o. R_{Co} is not partial ordering relation in general case.

In [1] it is said that the ordered folding set C_o is an implementable if the transitive closure of $R_{Co}{}^t$ of the ordering relation R_{Co} is a partial ordering relation on $R(B)$. Partially ordered set of rows that corresponds to R_{Co} can be embedded into a linearly ordered set of rows from $R(B)$. It is clear that implementable ordered folding set C_o contains all information needed to fold the matrix i.e. it specifies the pairs of columns to be folded together and their relative position (top or bottom). The objective is to find the implementable ordered folding set of maximum cardinality.

Example 2. There exist 7 folding pairs for matrix B corresponding to PLA on fig. 1: $P_u=\{c_1,c_2;\ c_7,c_8;\ c_1,c_4;\ c_1,c_5;\ c_2,c_3;\ c_2,c_5;\ c_5,c_6\}$.

The ordered folding set $<c_1,c_4>$ defines a partial ordering relation R_{14} of matrix B: $R_{14}=R(c_1) \times R(c_4)=\{(r_1,r_3), (r_1,r_4), (r_1,r_5), (r_2,r_3), (r_2,r_4), (r_2,r_5)\}$. For matrix B we see that $Co=\{<c_1,c_4>,<c_5,c_6>\}$ implies $R'_{Co}=R_{14}\cup R(c_5)\times R(c_6)=R_{14}\cup\{(r_4,r_1), (r_4,r_3), (r_4,r_5)\}$. Its transitive closure R_{Co} isn't a partial ordering relation on $R(B)$ because there exist pairs (r_4,r_1) and (r_1,r_4). Thus $\{<c_1,c_4>,<c_5,c_6>\}$ isn't implementable. In contradiction $\{<c_1,c_4>,<c_6,c_5>\}$ is implementable.

3. The Folding Problem Solution

In [5] the use of ordered folding pairs compatibility graphs for solving the folding problem is proposed. In such a graph $G=(V, E)$ the set of vertices V corresponds to the set of ordered folding pairs and an edge between two vertices v_i and v_j exists if and only if the appropriate ordered folding pairs are compatible, i.e. they give rise to implementable ordered folding set. It is shown that every implementable ordered folding set is a clique in G. The converse is not necessary true. Thus implementable ordered folding set of maximal cardinality is in the midst of dominant cliques of graph G.

Further in [4] for the sake of size reduction of a compatibility graph the idea is proposed to handle with unordered folding pairs instead of those ordered. That allows to reduce the number of vertices by one half and the number of edges up to $2n(n\text{-}1)$ times.

The marked unordered folding pairs compatibility graph $G=(V, E, M)$ is introduced. Here the nodes of V are one-to-one correspondence with the unordered folding pairs, an edge $e_i=(v_k,v_l)\in E$ and has a mark $m_i\in\{-,1,0\}$ if the there exists some ordering of appropriate unordered folding pairs $p_k=c_{k1},c_{k2}$ and $p_l=c_{l1},c_{l2}$ that generates compatible ordered folding pairs and m_i has a value:

- 1 if two orderings $<c_{k1},c_{k2}>,<c_{l1},c_{l2}>$ and $<c_{k2},c_{k1}>,<c_{l2},c_{l1}>$ are allowable (p_k and p_l can be ordered the same manner);
- 0 if two orderings $<c_{k1},c_{k2}>,<c_{l2},c_{l1}>$ and $<c_{k2},c_{k1}>,<c_{l1},c_{l2}>$ are allowable (p_k and p_l can be ordered the opposite manner);
- "-" (unmarked edge) if all of four orderings are allowable.

Example 3. The marked unordered folding pairs compatibility graph $G=(V, E, M)$ be induced by the set of unordered folding pairs P_u (example 2) is shown in fig. 2. Here the marks "-" of the graph's edges are omitted.

It has been shown [4] that the number of different permissible orderings unordered folding set is equal to the number of different permissible graph verteces' coloring.

The graph coloring means the following. The graph vertex color 1 means the appropriate

nordered pair c_k,c_l gives rise to the ordered folding pair $<c_k, c_l>$ (without permutation). Otherwise it gives rise to the ordered folding pair $<c_l, c_k>$ (with permutation). In fact once we have found graph coloring we have characterize an ordered folding set in graph terms. It would be said that for the purposes of PLA folding two graph colorings arising from each other by reversing the colors of all vertices are equivalent.

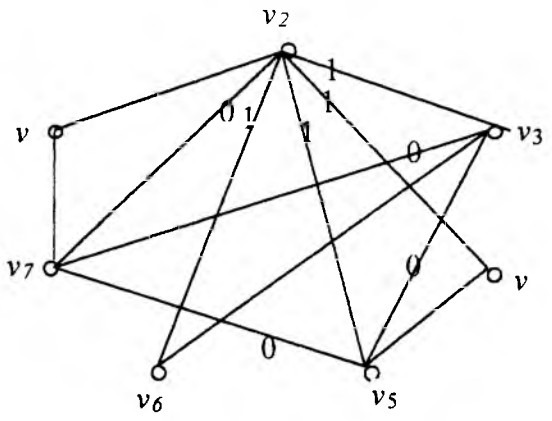

Figure 2. The marked graph $G=(V, E, M)$ of folding pairs' compatibility

The task of graph vertices' coloring is as follows. Two vertices of G connected with 1-marked edge should be colored the same manner (both in 1 or 0) and connected with 0-marked edge - the opposite manner. If two vertices are bound with unmarked edge there are no restrictions on their colors. Such a vertex coloring is called as permissible.

The number of different graph's colorings checked will be reduced if it is taken into account the following assertions from [4].

***Theorem* 1.** The marked graph $G=(V, E, M)$ cannot be colored in two colors if there exists a cycle satisfying to following conditions: 1) all its edges are marked (all marks are not "-"), 2) odd number of these edges are marked with 0.

***Theorem* 2.** If a cycle exists having the only unmarked edge this edge has to be marked so that the number of marked with 0 edges becomes even.

***Theorem* 3.** If the vertices of marked graph $G=(V, E, M)$ cannot be colored with two colors a cycle of three marked edges odd number of which marked with 0 will be met in the process of edges marking.

Example 4. Consider subgraphs of the marked graph $G=(V, E, M)$ (in fig. 2) with all marked edges and vertex sets $V_1=\{v_2, v_3, v_7\}$ and $V_2=\{v_2, v_3, v_5, v_7\}$ respectively. The first of these graphs has even number (2) of 0 marked edges. Thus its vertices can be colored with two colors. The second one has odd number (3) of 0 marked edges. Thus it cannot be colored with two colors.

The search for the cycle mentioned in the Theorem 1 can be combined with the process of marking unmarked edges. This edges' marking would be done by the matching the theorems mentioned above.

In [4] it is shown that an implementable ordered folding set under discussion is among the dominant cliques that has a permissible vertex coloring of the marked graph $G=(V, E, M)$ and we have as many permissible ordered folding sets as permissible vertex colorings exist. Thus the implementable folding set of maximum cardinality is in the midst of dominant cliques of the marked graph $G=(V, E, M)$ allowing permissible vertex coloring.

4. The Algorithm of Folding Problem Solution

We are going to build successively one by one cliques of the marked graph $G=(V, E, M)$ allowing permissible vertex colorings. After getting a new better solution we store it and the respective number n of its vertices.

At every next step we need to consider only those cliques G^m_t with more vertices than that early founded one ($m>n$). The clique G^m_t under consideration at each step should be examined for existence at least one of the ordered folding sets, though it can give rise to as more ordered folding sets as the number of different colorings (in two colors) graph vertices is. In this step we find subsequently one by one the ordered folding sets and test them additionally for implementability until we find implementable ordered folding set or come to an end. If we get a new better solution we can reduce graph G at the sacrifice of elimination of all vertices with degrees less than n.

The search tree. The set of all possible cliques is organized in the form of a search tree in which each node represents a set of clique vertices of the graph. The sons of a node represent those cliques that contain one additional vertex. By traversing this search tree all possible cliques can be systematically examined.

A node d_k of the search tree presents a set V_k^r of clique vertices and it is represented by a sub-graph $G_k=(V_k, E_k, M_k)$ of the graph $G=(V, E, M)$. This sub-graph contains the candidates for adding into a clique searched. So, $V_k=V_k^r \cup (\cap \Gamma v_i / v_i \in V_k)$: $E_k=\{e=(v_i,v_j)/(v_i,v_j \in V_k)\&((v_i,v_j)\in E)\}$.

Restricting the search tree. A node d_k of the search tree is called viable if:

1. the cardinality of its set V_k is greater than the cardinality of the dominant clique founded when traversing the tree before getting to this node;
2. the graph $G_k=(V_k, E_k, M_k)$ has some permissible vertex coloring [4];
3. there exists ordered folding set induced by vertex coloring that is implementable.

These conditions are ordered according to their complexity. The test for ordered folding set implementability exceeds the limits of this report.

Only viable nodes need to be examined during an exhaustive search of the tree. The sons of a node represent those cliques that can be derived from their father by adding one new vertex.

Heuristic for choosing the traversing path of the tree. The algorithm suggested employs a simple heuristic to determine an order in which sub-trees rooted in a node are traversed so that a near optimal solution can be discovered quickly. Finding a near-optimal solution quickly is important since sooner such a clique is found the sooner it can be used for pruning purposes.

It is proposed to alter dynamically the structure of the search tree by choosing what son to visit first at each node under consideration. A simple criterion is used - the degree of the vertex $v_i \in V_k$ to be added to a current clique set V^r_k: it is preferable to choose a vertex with the greatest local degree in graph G_k.

5. THE SEARCH ALGORITHM

Traversal of the search tree is done in a depth-first manner, backtracking from a node whenever the sub-tree rooted by a node has been completely explored. The size of such search trees grows exponentially. However the suggested branch and bound algorithm restricts the exploration to within only a small subset of the nodes in the tree. This elimination of nodes in the search tree is called pruning search tree, and can significantly reduce the execution time of the search algorithm. A sub-tree can be pruned only if the algorithm can determine that this sub-tree contains no ordered folding set that is larger in size than largest ordered folding set that has been found so far.

The algorithm upon visiting a node, transforms the graph G_k for the node's father d_k into the G_{k+1} for the current node's son d_{k+1}.This transformation is induced by including into a set V_k^r a new vertex

$$v_l \in V_k: V_{k+1}=v_l \cup V_k^r \cap \Gamma v_l;$$

$$E_{k+1}=E_k \cap \{e=(v_i,v_l)/(v_i \in V_k)\&((v_i,v_l) \in E_k)\}.$$

If a new node is viable the algorithm proceeds otherwise it backtracks above in the search tree.

When backtracking from the son's node d_{k+1} to father's node d_k the graph G_k is reduced:

1. the vertex v_l connected the nodes d_k and d_{k+1} is deleted from V_k together with its incident edges;
2. a vertex with degree no greater than the number n of vertices of a dominant clique found is excluded from V_k too.

If the cardinality of V_k becomes not greater than n the algorithm in turn returns to fathers node d_{k-1}.

6. THE PROCEDURE OF VERTEX COLORING

To test the current graph G_k for existence of permissible vertex coloring it is proposed to perform the partial necessary coloring vertices from V_k. The partial coloring partitions the vertex set into two sub-sets: those colored and those uncolored. That partially colored graph induces ordered folding pairs and unordered folding pairs. That is there exist several permissible ordered folding sets.

The algorithm is based on the Theorems 1-4 proved in [4].

Theorem 4. Let U_1^r и U_0^r are sets of vertices that are connected in G_k with $u_m \in V_k^r$ with 1- and 0-marked edges correspondingly. There exists no permissible coloring of vertices of the graph $G^r_k=(V_k^r, E_k^r, M_k^r)$ even if one of the following conditions is not satisfied:

1. all colored vertices from U_1^r are p-colored ($p \in \{1, 0\}$) and all colored vertices from U_1^r are 'p-colored ('p is an inversion of p);
2. if u_m is p_m-colored, then $p_m=p$ (if there are any colored vertices in U_1^r and/or U_0^r)

When the algorithm proceeds to the node d_{k+1} from d_k by adding a vertex v_l into V_{k+1}^r we shell use the set U of vertices to analyze. Initially U contains v_l only. Then for every $u_m \in U$ we question if the conditions of the theorem stated above are satisfied. If they are not we have to backtrack because we have no implementable ordered folding set in this sub-tree. Otherwise we pick out the sub-sets U_1 и U_0 of vertices from V_k that are connected with u_m with 1- and 0-marked edges correspondingly.

Then if even one of the vertices from $U_1 \cup U_0 \cup u_m$ is colored all permissible colorings uncolored vertices from $U_1 \cup U_0 \cup u_m$ should be done: the vertices from $U_1 \cup u_m$ must be colored

the same manner and the vertices from U_0 - the opposite manner. The vertices that cannot be colored so must be deleted from V_k - the subset of candidates for including into ordered folding set. All just now colored vertices are included in U for the purposes of further analysis.

Example 5. Let us find an ordered folding set of maximum cardinality for Boolean matrix from example 1. It is in the midst of dominant cliques of the marked graph $G=(V, E, M)$ of folding pairs' compatibility (fig. 2) allowing permissible vertex colorings.

Below we list the course of computation when seeking a maximal dominant clique that corresponds to ordered folding set of interest. At each step of the algorithm flow we show

1. the current analyzed node d_k of the search tree (in fact k is the current search tree depth),
2. the current states of graphs G_k, G_k^r by means listing their vertex sets V_k and V_k^r,
3. the current set M_k of graph vertex colors and
4. the vertex chosen for including in G_{k-1}^r (with its degree in brackets):

d_1: $V_1=\{v_1,v_2,\ldots,v_7\}$, $V_1^r=\varnothing$, $M_1=\{-,-,\ldots,-\}$, v_2 (6);
d_2: $V_2=\{v_1,v_2,v_3,v_4,v_5,v_6,v_7\}$, $V_2^r=\{v_2\}$, $M_2=\{-,1,1,1,1,1,0\}$, v_7 (4);
d_3: $V_3=\{v_1,v_2,v_3,v_5,v_7\}$, $V_3^r=\{v_2,v_7\}$, $M_3=\{-,1,1,1,0\}$, v_3 (3);
d_4: $V_4=\{v_2,v_3,v_7\}$, $V_4^r=\{v_2,v_7,v_3\}$, $M_4=\{1,1,0\}$.

When proceeding from father node d_3 to son node d_4 (after the vertex v_3 has been chosen) the vertex v_5 is excluded from V_4. That is why v_3 is 1 colored results in $v_5 \in U_0$ but v_5 is 1 colored (as a result of v_2 has been 1 colored). Thus the search tree node d_4 is declared as leaf node. There exists a maximal clique corresponding to this node and having permissible vertex coloring. This clique has 3 vertices: $n=3$.

When backtracking from the node d_4 we see the search tree traversals from all nodes lead to leaf nodes. That means there does exist no clique with more than three vertices:

d_3: $V_3=\{v_2, v_5, v_7\}$, $V_3^r=\{v_2, v_7\}$, $M_3=\{1, 1, 0\}$;
d_2: $V_2=\{v_2, v_5\}$, $V_2^r=\{v_8\}$, $M_2=\{1, 1\}$;
d_1: $V_1=\varnothing$, $V_1^r=\varnothing$, $M_1=\varnothing$.

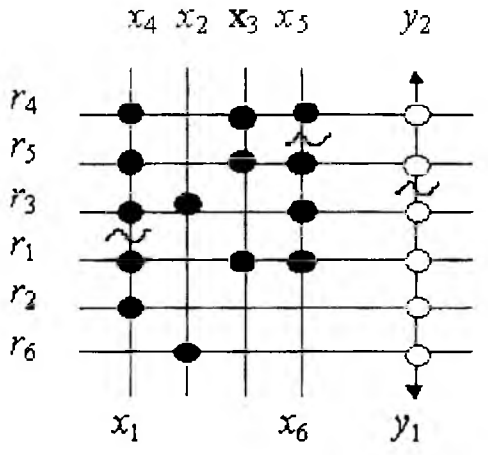

Figure 3. Column folded PLA matrix

Thus we have found one of the maximal dominant cliques having a permissible vertex coloring. It has three vertices: $\{v_2, v_7, v_3\}$. The ordered folding set corresponding to this clique is $Co=\{<c_8,c_7>, <c_4,c_1>, <c_5,c_6>\}$. It is implementable, the associated with it column folded PLA is shown in fig. 3.

REFERENCES

[1]. Hachtel G.D., Newton A.R., Sangiovanni-Vincentelli A.L. An Algorithm for optimal PLA Folding. - *IEEE Trans. Comput.-Aided Design of ICAS*, v. CAD-1, 1982, **2**, pp. 63-77.

[2]. De Micheli G., Sangiovanni-Vincentelli A. Multiple Constrained Folding of Programmable Logic Arrays: Theory and Applications. - *IEEE Trans. Comput.-Aided Design*, - v. CAD-2, 1983, **3**, pp. 151-167.

[3]. Grass W. A Depth-first branch-and-bound algorithm for optimal PLA folding. - Proceedings of *19th Design Automation Conf. Proc.*, Las Vegas, June 14-16, 1982, pp. 133-140.

[4]. Cheremisinova L.D. Some results in optimal PLA folding. - Proceedings of the *International Conference on Computer-Aided Design of Discrete Devices* (CAD DD'99), v. 1, Minsk, 1999, pp. 59-64.

[5]. Lecky J.E., Murphy O.J. and Absher R.G. Graph theoretic algorithms for the PLA folding problem.- *IEEE Trans. Comput.-Aided Design*, - v. 8, 1989, **9**, pp. 1014-1021.