OPTIMIZING PROGRAMMABLE LOGIC ARRAYS USING THE SIMULATED ANNEALING ALGORITHM

Liudmila Cheremisinova, Irina Loginova

***Abstract**: In the paper the programmable logic array (PLA) topological optimization problem is dealt with using folding techniques. A PLA folding algorithm based on the method of simulated annealing is presented. A simulated-annealing PLA folding algorithm is presented for multiple unconstrained folding. Then, the algorithm is extended to handle constrained folding. In this way, simple folding is considered as a case of multiple constrained folding. Some experimental results of computer investigation of the suggested algorithms are given.*

***Keywords**: design automation, area optimization, VLSI structure folding, simulated annealing.*

***ACM Classification Keywords**: B.6.3 [Logic Design]: Design Aids – Optimization; B.7.2 [Integrated circuits]: Design Aids –Layout.*

Introduction

Structured logic refers to logic forms that exhibit a high degree of regularity in their layout and interconnections. Such layout is referred to a regular layout style that is constructed according to some definite architecture and established in advance for some structured logic. Typical widely used regular structures are Programmable Logic Arrays (PLA's), gate matrix arrays, Weinberger Arrays [Ullman, 1984]. All these forms have a two-dimensional structure consisting of a matrix 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. In order one to have a clear idea of degree of structural rarity, it can be said that previous research has shown that on average, about half of the entries in the personality matrices of large structures in real circuits are not personalized. Several techniques have been proposed for reducing the area required.

Since matrix is the central part of any regular structure mentioned above, we consider further just it and then we show how to expand the obtained results to real regular structures taking into account the peculiarities of their layout and some other constraints resulted from the implementation circuit on the base of these array structures and the chosen type of topological minimization.

Techniques of topological minimization reduce the number of physical columns and/or rows, they change the PLA structure by using a procedure called folding [Hachtel, 1982], [DeMicheli, 1983]. Folding is a technology-independent transformation, it is developed for array structures and attempts to place two or more columns (and/or rows) together in the same physical vertical (and/or horizontal) line (signal bus) so that they can share this line. Column folding is said to be simple if utmost two columns (rows) share a single physical vertical (and/or horizontal) line. It is called multiple if more than two columns can share a single column.

Folding does not change the implemented logic in any manner, but reduces the number of columns (and/or rows), and thus reduces the area. In the paper we deal with array structure folding and its effects on its square.

[Wong, 88] proposes the use of simulated-annealing to solve the PLA folding problem for the case of multiple unconstrained folding. In this approach, entire solutions are analyzed one after another by producing different permutations of rows. In the paper we also propose the solution of multiple unconstrained and constrained folding problem and extend the suggested simulated-annealing algorithm to solve a special case of multiple constrained folding – simple folding. Then we give some experimental results of computer investigations of the suggested simulated-annealing algorithms based on simple and multiple PLA folding and comparative evaluation of their effects on PLA layout reduction.

Array structures style and their folding

Any array structure can be represented by specifying the positions of of their elements (transistors) in its plane. In the folding problem it can be described in symbolic form by a Boolean matrix $\boldsymbol{B}$ having sets $C(\boldsymbol{B})$ and $R(\boldsymbol{B})$ of their columns (where uncomplemented and complemented modes of a variable have their own distinct column) and rows. A 1 in the position i,j of the matrix $\boldsymbol{B}$ means that there is an appropriate crosspoint (transistor) between the i-row and the j-column in the matrix. Each column $c_i \in C(\boldsymbol{B})$ implies a set $R(c_i) \subseteq R(\boldsymbol{B})$ of rows, which are populated on it: $r_j \in R(c_i) \leftrightarrow b_j^i = 1$. For instance, for PLA on Figure 1 Boolean matrix $\boldsymbol{B}$ corresponding to AND-plane is depicted in Figure 2.

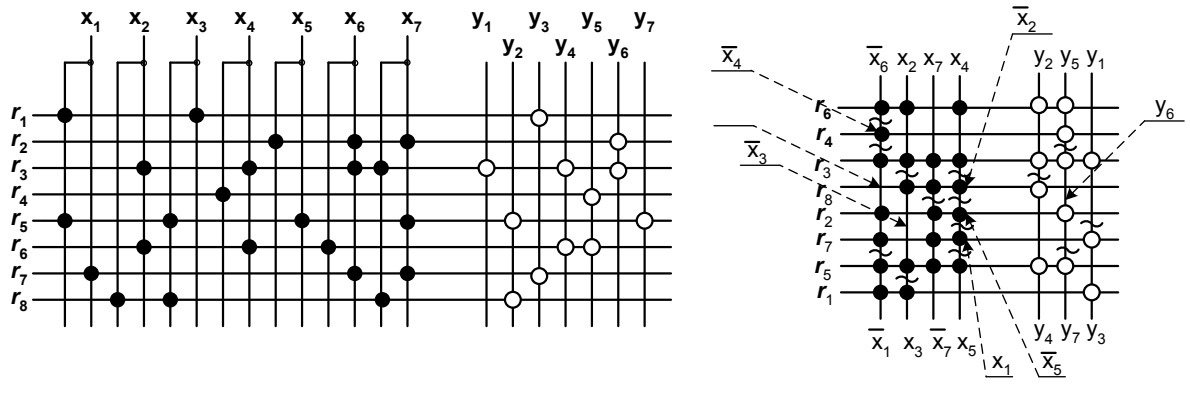

Figure 1. An example of array-based structure: a) PLA; b) PLA multiple folded form

	$\bar{x}_1$ c_1	x_1 c_2	$\bar{x}_2$ c_3	x_2 c_4	$\bar{x}_3$ c_5	x_3 c_6	$\bar{x}_4$ c_7	x_4 c_8	$\bar{x}_5$ c_9	x_5 c_{10}	$\bar{x}_6$ c_{11}	x_6 c_{12}	$\bar{x}_7$ c_{13}	x_7 c_{14}
r_1	1	0	0	0	0	1	0	0	0	0	0	0	0	0
r_2	0	0	0	0	0	0	0	0	1	0	0	1	0	1
r_3	0	0	0	1	0	0	0	1	0	0	0	1	1	0
r_4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
r_5	1	0	0	0	1	0	0	0	0	1	0	0	0	1
r_6	0	0	0	1	0	0	0	1	0	0	1	0	0	0
r_7	0	1	0	0	0	0	0	0	0	0	0	1	0	1
r_8	0	0	1	0	1	0	0	0	0	0	0	0	1	0

Figure 2. Boolean matrix ***B*** reflecting the structure of PLA AND plane of Figure 1, a

The goal of the array structure folding is to find the maximum number of columns/rows that can be folded simultaneously.

Any two columns c_i and c_j are disjoint if $R(c_i) \cap R(c_j) = \varnothing$. Two disjoint columns both do not have transistors on any particular row of the array-based structure. A column folding list $l^c_k = (c_{k1}, c_{k2},\ldots, c_{km})$ is a set of pairwise disjoint columns c_{ij}, it is unordered in general case. An ordered column folding list $l^{co}_k = <c_{k1}, c_{k2},\ldots, c_{km}>$ is a column folding list l^c_k whose elements c_{ki} are ordered. An ordered column folding list (OCFL) l^{co}_k of cardinality two is an ordered column folding pair. Any OCFL $l^{co}_k = <c_{k1}, c_{k2},\ldots, c_{km}>$ can be actually implemented in the same vertical line of array-based structure moving c_{k1} above c_{k2}, c_{k2} above c_{k3}, an so on, $c_{k,m-1}$ above c_{km}. So OCFL l^{co}_k results to permutation on the set of rows: $R(c_{k1}) > R(c_{k2})$ – the rows of $R(c_{k1})$ are all above those of $R(c_{k2})$, $R(c_{k2}) > R(c_{k3})$ – $R(c_{k2})$ are all above those of $R(c_{k3})$, and so on, $R(c_{k,m-1}) > R(c_{k,m})$ – the rows of $R(c_{k,m-1})$ are all above those of $R(c_{km})$ inducing the relation on row set $R(\boldsymbol{B})$:

$$P^r(l^{co}_k) = \bigcup_{i,j} (R(c_{ki}) \times R(c_{kj})), \quad \text{i.e.} \quad P^r(l^{co}_k) = \{ r_p \times r_q / r_p \in R(c_{ki}), r_q \in R(c_{kj}), i < j \}.$$

This relation is partial because it is irreflexive, asymmetric and transitive by its definition.

An ordered column folding set (OCFS) $L^{co} = \{l^{co}_1, l^{co}_2,\ldots, l^{co}_k\}$ is a set of disjoint ordered column folding lists. The number k of columns entering into all OCFLs $l^{co}_i \in L^{co}$ is called the size of OCFS L^{co}. OCFS L^{co} induces a set of ordering relations among the rows that is the union of ordering relations induced by OCFLs l^{co}_k belonging to the OCFS $P^r(l^{co})$:

$$P^r(L^c_k) = \bigcup_{i=1}^{k} (P^r(l^{co}_i)).$$

This relation $P^r(L^{co})$ is irreflexive, asymmetric but not transitive in general case. The transitive closure $R^t(P^r)$ of $P^r(L^{co})$ is irreflexive, transitive but can be not asymmetric.

It is proven [DeMicheli, 1983] that an OCFS L^{co} is implementable topologically (by a folded array-based structure) if the transitive closure $R^t(P^r)$ of the relation $P^r(L^{co})$ is a partial ordering on $R(\boldsymbol{B})$, that is $R^t(P^r)$ is asymmetric. In other

words, an OCFS $L^{co}{}_k$ is implementable topologically if there exist linear order of the rows $R(\boldsymbol{B})$ extending the row ordering $P^r(L^{co})$.

An implementable OCFS L^{co} specifies the structure of the folded array, and its size is referred to as the size of the folding: the number of OCFLs in L^{co} corresponds to the number of columns that will replace $\sum_{i=1}^{k} |l^c_i|$ columns of the initial array-based structure.

An example of a PLA structure (in pictorial symbol) and its multiple folded form is depicted in Figure 1. Here the "breaks" of the lines that occur on the folded columns are designated by the symbol "~". The size of implementable OCFS L^{co} = {< $\bar{x}_6$, $\bar{x}_4, x_6$, $\bar{x}_1$>, <x_2, $\bar{x}_3, x_3$>, <x_7, $\bar{x}_7$>, <x_4, $\bar{x}_2$, $\bar{x}_5, x_1, x_5$>, <y_2, y_4>, <y_5, y_6, y_7>, <y_1, y_3>} (corresponding to the folded PLA) is equal seven and thus 7 columns of the folded regular structure replace 21 columns of the initial structure.

So the formal statement of optimal folding problem is as follows: given a Boolean matrix representing array-based structure, find an implementable ordered folding set of maximum size.

Simulated annealing formulation

Simulated annealing is a computer approach widely used to solve difficult optimization problems. Such an approach is applied to a wide variety of applications where the search for optimal solution is needed. Any problem requiring optimal or near optimal solution over a space formed from the combination of several variables is considered to be a combinatorial approximation problem of the following type [Lee, 1995]:

$$\textit{Minimize (maximize)}\ f(x), \text{ subject to } g_j(\boldsymbol{x}) = 0,\ j = 1, 2,\ldots, \tag{1}$$

where $f(x)$ is the cost (or object) function over the vector of configuration variables x_i, and $g_j(x)$ are some constraints.

The first widely available publication on simulated annealing belongs to Kirkpatrick et al. [Kirkpatrick, 1983]. Simulated annealing, developed by Kirkpatrick et al. treats combinatorial approximation analogously to the annealing of metals. As in the actual annealing process (in which a good crystal structure is desired as the final result), simulated annealing requires a carefully controlled cooling schedule to avoid a bad final solution. The basic algorithm has been thoroughly discussed over the past several years (their overview could be found in [Lee, 1995], [Greening, 1995]).

In general case the combinatorial optimization problems could be presented as follows: there is a finite set G of states $g_i(\boldsymbol{x})$ where each state $g_i(\boldsymbol{x}) \in G$ is represented by n state variables: $g_i(\boldsymbol{x}) = (x_1, x_2,\ldots, x_n)$. Then a cost function Cost($g_i(\boldsymbol{x})$) is given. The goal is to find out the state $g_i(\boldsymbol{x})$ with minimum cost. Overwhelming majority of such problems are NP-complete, so the algorithms to solve such problems require exponential time relative to n. A near optimal state is often good enough in practice. One of the polynomial time heuristics for these problems is

the greedy algorithm. It does not always produce a satisfactory outcome but it is good algorithmic basis for simulated annealing.

Simulated annealing augments the greedy algorithm with a random escape from local minima converting it to a probabilistic hill-climbing algorithm. The escape is controlled through a value called as "temperature". Higher temperature makes the algorithm more likely to increase cost when selecting a trial state. In this way simulated annealing can climb out of a local minimum. Figure 3 [Greening, 1995] shows how a simulated annealing algorithm works.

$t \leftarrow t_s$;
$g \leftarrow$ starting state;
$Cost \leftarrow Cost(g)$;
while not stopping criteria( )
    $g^* \leftarrow$ generate(g) with probability G_{ss^*};
    $Cost^* \leftarrow Cost(g^*)$;
    $\triangle \leftarrow Cost^* - Cost$;
    if $(\triangle \leq 0) \vee (\text{random}() < e^{-\triangle/t})$
        $g \leftarrow g^*$;
        $Cost \leftarrow Cost^*$;
    $t \leftarrow$ reduce temperature(t);
end while

Figure 3. Simulated annealing procedure

The first three operators of the simulated annealing procedure (Figure 3) set the initial temperature t, the current state g and its cost $Cost$. The loop generates a trial state g^*, evaluates its cost $Cost^*$ and change of cost $\triangle$. Then if the condition is satisfied the algorithm selects this new state g^* as the next current state g and reduces the temperature until the stopping criteria is met. The condition "if $(\triangle \leq 0) \vee (\text{random}() < e^{-\triangle/t})$" shows how simulated annealing accepts a trial state. The term $(\triangle \leq 0)$ expresses greedy strategy (it chooses a lower cost trial state). The function random() returns a uniformly distributed random value between 0 and 1. The term $(\text{random}() < e^{-\triangle/t})$ evaluates the likelihood of permitting a costlier trial state, the probability of accepting a costlier trial state decreases exponentially with the increase in cost and the decrease in temperature t.

The function "reduce temperature (t)" decreases the temperature according to a cooling schedule, which provides fulfilling four major tasks in the proper way:

1) to set high enough starting temperature to accept most of the moves;

2) to determine when the present temperature is to be decreased;

3) to determine the next temperature;

4) to finish the process.

Together, the methods of generating moves and the cooling schedule form the foundation of simulated annealing procedure. A starting temperature t_0 must be assigned or calculated and a series of random alterations (iterations) is then made. In accordance with the accepted condition of moves the sequence of states at one temperature forms a Markov chain. When the chain reaches equilibrium at a particular temperature, the temperature is decreased according to some cooling schedule and the procedure is repeated. The algorithm terminates when a specified stopping condition is reached.

The number of iterations in an annealing run appears to be the most critical parameter in annealing, many methods have been suggested (their review cold be found in Lee, 1995]). The simplest of them is the geometric cooling schedule, where the starting temperature t_s, chain length L, temperature decrement, and stopping temperature T_e are all predefined.

Multiple folded regular structure realization

Column folding can be obtained by permuting the rows of an array structure, so it introduces a restriction on the order of the rows. That is, if column c_i is folded with column c_j and placed above it, then all the rows that have cross points with the column c_i must be placed above those rows that have cross points with the column c_j. For example, in Figure 1, to fold column x_6 above $\bar{x}_1$, thus rows r_2, r_3 and r_7 should be placed above r_1 and r_5.

The restrictions imposed on the order of rows by a list of folded columns can conflict with the row ordering desired by another list of folded columns. An unconflicting ordering of rows is decided as optimal if it needs to the maximal folding of columns or the minimal number of physical vertical lines. So we should find out such an unconflicting ordering of rows.

	$\bar{x}_1$ c_1	x_1 c_2	$\bar{x}_2$ c_3	x_2 c_4	$\bar{x}_3$ c_5	x_3 c_6	$\bar{x}_4$ c_7	x_4 c_8	$\bar{x}_5$ c_9	x_5 c_{10}	$\bar{x}_6$ c_{11}	x_6 c_{12}	$\bar{x}_7$ c_{13}	x_7 c_{14}
r_6	0	0	0	1	0	0	0	1	0	0	1	0	0	0
r_4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
r_3	0	0	0	1	0	0	0	1	0	0	0	1	1	0
r_8	0	0	1	0	1	0	0	0	0	0	0	0	1	0
r_2	0	0	0	0	0	0	0	0	1	0	0	1	0	1
r_7	0	1	0	0	0	0	0	0	0	0	0	1	0	1
r_5	1	0	0	0	1	0	0	0	0	1	0	0	0	1
r_1	1	0	0	0	0	1	0	0	0	0	0	0	0	0

Figure 4. Boolean matrix $\boldsymbol{B}^{\pi}$ induced by the matrix $\boldsymbol{B}$ of Fig.2

	$\bar{x}_1$ c_1	x_1 c_2	$\bar{x}_2$ c_3	x_2 c_4	$\bar{x}_3$ c_5	x_3 c_6	$\bar{x}_4$ c_7	x_4 c_8	$\bar{x}_5$ c_9	x_5 c_{10}	$\bar{x}_6$ c_{11}	x_6 c_{12}	$\bar{x}_7$ c_{13}	x_7 c_{14}
r_6	0	0	0	**1**	0	0	0	**1**	0	0	**1**	0	0	0
r_4	0	0	0	1	0	0	**1**	1	0	0	0	0	0	0
r_3	0	0	0	**1**	0	0	0	**1**	0	0	0	**1**	**1**	0
r_8	0	0	**1**	0	**1**	0	0	0	0	0	0	1	**1**	0
r_2	0	0	0	0	1	0	0	0	**1**	0	0	**1**	0	**1**
r_7	0	**1**	0	0	1	0	0	0	0	0	0	**1**	0	**1**
r_5	**1**	0	0	0	**1**	0	0	0	0	**1**	0	0	0	**1**
r_1	**1**	0	0	0	0	**1**	0	0	0	0	0	0	0	0

Figure 5. Interval matrix $\boldsymbol{L}^{\pi}$ corresponding to $\boldsymbol{B}^{\pi}$ in Fig. 4

As column folding is formulated as a row permutation problem, during the column folding phase the rows of an array structure are permuted to ensure some columns could be share the same vertical lines of the folded structure using less lines comparing the initial number of columns. To derive the conditions row permutation should satisfy to lead to a maximal size column folding, let consider the result of multiple folding of regular

structure (Figure 1). It corresponds to some row permutation π that defines the row sequence and is one-to-one function of assigning rows of initial matrix $\boldsymbol{B}$ (Figure 2) to horizontal lines: $\{\pi(r_1), \pi(r_2)..., \pi(r_m)\} \rightarrow \{r_6, r_4, r_3, r_8, r_2, r_7, r_5, r_1\}$. So the Boolean matrix $\boldsymbol{B}$ will be transformed into the Boolean matrix $\boldsymbol{B}^{\pi}$ (Figure 4) with permuted rows that is functionally equivalent to $\boldsymbol{B}$ but correspond the other structure realization. Then, the columns of $\boldsymbol{B}^{\pi}$ will be rearranged in top-to-bottom manner and a column partition is resulted describing a folded structure in symbolic form. The matrix $\boldsymbol{B}^{\pi}$ defines a graph $G = (V, E)$, where V is the set of columns and two vertices v_i and v_j are connected by an edge $e_{ij} \in E$ if the appropriate columns c_i^{π} and c_j^{π} intersect (have a 1 in the same row): $c_i \cap c_j \neq \varnothing$.

An ordering of rows induced by permutation π is termed as optimal if it induces the maximal folding of columns or, as the same, the minimum number of physical vertical lines of the folded structure. It follows from the folded structure that there exists column partition $C^{\pi} = \{C_1, C_2,..., C_k\}$ concerning to the row permutation π, such that all the columns of each component set C_k are mutually compatible and so form the folding list, and C^{π} as a folding set is realizable. For our example, for the Boolean matrix $\boldsymbol{B}^{\pi}$ we could obtain $C^{\pi} = \{\{c_{11}, c_7, c_{12}, c_1\}, \{c_4, c_5, c_6\}, \{c_{13}, c_{14}\}, \{c_8, c_3, c_9, c_2, c_{10}\}\}$. The cardinality k of the partition C^{π} defines the width of the row-permuted folded structure or the number of packed columns in it. The goal is to find out a row permutation that permits a column partition C^{π} of minimal cardinality.

Now let us introduce some notions to formulate the procedure of evaluation of a row permutation π. For a given permutation π the regular structure folding is realized by attaching to each vertical line column parts sharing it. Such parts correspond to intervals [introduced by Wong, 1988] that are established in the columns of the matrix $\boldsymbol{B}^{\pi}$ from the topmost to the lowermost 1's: $\boldsymbol{I}_k = [\min_i c_i^k, \max_i c_i^k,]$, where c_i^k is i-th component of k-th column having value 1. Such a way the matrix $\boldsymbol{B}^{\pi}$ induces interval matrix $\boldsymbol{L}^{\pi}$ for a given row permutation π. Just the intervals will be assigned in the proper way to vertical lines of the folded array structure also need the minimum number of vertical lines for the folded structure. When proceeding from $\boldsymbol{B}^{\pi}$ to $\boldsymbol{L}^{\pi}$ the densities d^{π}_i of the rows (the number of columns intersecting the row d^{π}_i) are increased in the general case. Reducing the row densities promotes placing more columns on a single physical line, and that can be done by proper permutation of rows.

For our example, we have the interval matrix $\boldsymbol{L}^{\pi}$ shown in Figure 5 (where the initially given 1's are represented in thick print in contrast to augmented 1's that are of regular type) and following the collection of intervals:

$$L^{\pi} = \{[7,8], [6,6], \{[4,4], \{[1,3], \{[4,7], \{[8,8], \{[2,2], \{[1,3], \{[5,5], \{[7,7], \{[1,1], \{[3,6], \{[3,4], \{[5,7]\}.$$

To find out the size of folding associated with the permutation π let consider the compatibility relation between intervals: a pair of intervals $\boldsymbol{I}_i$, $\boldsymbol{I}_j$ are compatible if they do not intersect (the appropriate columns of the matrix $\boldsymbol{L}_{\pi}$ do not have a 1 in the same row): $\boldsymbol{I}_i \cap \boldsymbol{I}_j = \varnothing$. Compatible intervals may be assigned the same vertical line in a proper layout. The inverse relation is the incompatibility relation that can be represented by the intersection graph

$G = (V, E)$, where the vertices of V correspond to intervals and $(v_i, v_j) \in E$ if and only if $\boldsymbol{l}_i$ and $\boldsymbol{l}_j$ intersect, ($\boldsymbol{l}_i \cap \boldsymbol{l}_j \neq \varnothing$). The graph is an interval graph. Interval graphs are a kind of the chordal graphs [Hsu, 1999] having some remarkable properties reducing NP-complete problem of searching of realizable folding sets to a solvable in polynomial-time.

It have been stated that the size of folding (associated with row permutation π) equals to the chromatic number of an interval graph $G = (V, E)$ and the color classes correspond to collections of intervals each of which can be feasibly assigned to a line. Then the chromatic number of the interval graph is equal to the maximal degree of vertices of the graph G. In other words the size of folding associated with a row permutation π can be found directly from the corresponding interval matrix $\boldsymbol{L}^{\pi}$ – it is the maximal density of the matrix rows (denoted later as d^{π}) [Wong, 1988]. And further a minimum size folding induced a row permutation π can be found solving 1) the task of graph coloring: the color classes of an interval graph may be obtained by applying a simple linear-time algorithm to the vertices of an interval graph $G = (V, E)$ [Golumbic, 1977] , or 2) the task of interval ordering by applying to the collection L^{π} of intervals a special simple "Left Edge Algorithm" [Hashimoto, 71].

The interval graph $G = (V, E)$ for our example is depicted in Figure 6. The graph chromatic number equals 4. The appropriate ordered column folding set is $L^{co} = \{<c_{11}, c_7, c_{12}, c_1>, <c_4, c_5, c_6>, <c_{14}, c_{13}>, <c_8, c_3, x_9, c_2, c_{10}>$, it is implementable and the corresponding folded regular structure is given as a part of Figure 1, b.

Thus from above, it can draw a conclusion that any row permutation is evaluated in a simple way: the value of d^{π} is the numerical value of the row permutation quality. And after choosing the best row permutation, the partition of the columns set on a collection of column folding lists is fulfilled by linear-time algorithm.

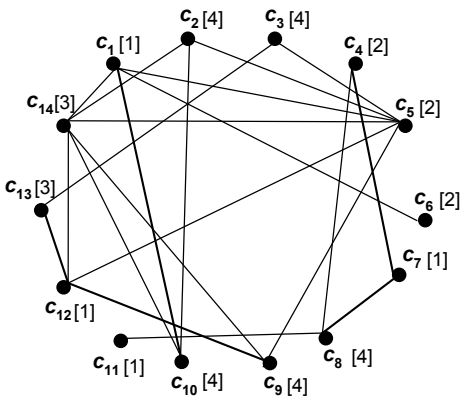

Figure 6. Interval graph $G = (V, E)$

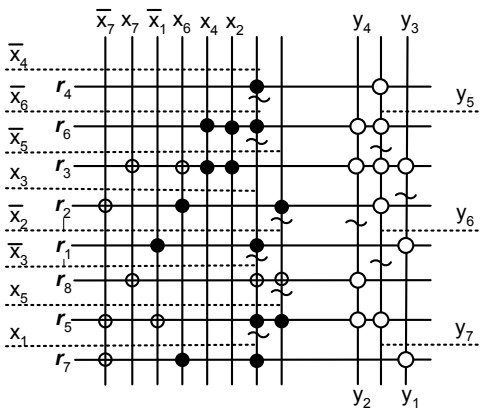

Figure 8. PLA multiple folded form subject to signal routing lines

Multiple folding via the method of simulated annealing

Here we consider how to solve the problem of multiple column folding using the method of simulated annealing. During the column-folding phase, the rows of an array structure are permuted to minimize the area due to folding some columns. In that case array structure will have fixed height and decreased width. The aim of the column-folding phase is to minimize the width of the structure. Indirectly, to minimize the width, it would be necessary to minimize the number of the columns. The goal is to obtain an ordering of rows that will allow maximal packing the columns.

Figure 7 shows how the realized simulated annealing folding algorithm works. The current row order π in the matrices $\boldsymbol{B}^{\pi}$ and $\boldsymbol{L}^{\pi}$ is a state in the process of simulated annealing. And the starting state is the initial row order $\{r_1, r_2, \ldots, r_m\}$. A move is a permutation of a pair of columns and it changes the current state for some neighbor one. The temperature schedule is represented by geometric series where starting value $t_0 = \lambda_t n$ and each successive temperature value is obtained by multiplying the previous one by some constant value that is 0.85 that is less than 1 and so the temperature values approach zero in the limit. Here n is the number of columns, the value $\lambda_t < 1$ is taken as 0.8 (for it experimental results indicate good results obtained) [Wong, 1988]. The number of iterations (moves) at each temperature is accepted as $r_0 = 2\lambda_r C_m^2$, $C_m^2 = m(m-1)/2$ is the number of 2-combinations (the number of possible neighbor states) – the number of all possible moves where the value $\lambda_r < 1$ is taken as 0.5 [Van Laarhoven, 1987], m is the number of rows.

Each trial state is evaluated with a cost function Q^{π}. The goal of an annealing folding algorithm is to find out the interval matrix $\boldsymbol{L}^{\pi}$ with minimum of maximal density: $\max of\ d^{\pi} \to \min$. Taking into account that the row permutation and the corresponding to it maximal density are random quantities, the cost function Q^{π} is expressed as quadratic one:

$$Q^{\pi} = (d^{\pi})^2 + \frac{\lambda_{\pi}}{m}\sum_i (d_i^{\pi})^2 , \qquad (2)$$

where λ is accepted to be 0.5 reducing the influence of extra item $\frac{1}{m}\sum_i (d_i^{\pi})^2$ by a half. The trial is successful if the cost value of the current state is less than that of the best previous state. The simulated annealing process is terminated if the number of successful steps at a current temperature is less than 5% of all moves or the temperature became low enough.

$\boldsymbol{B} \leftarrow \boldsymbol{B}_0$ folding	$\boldsymbol{B}_0$ – initially given Boolean matrix presenting regular structure under
$\pi \leftarrow \pi_0$	π_0 – initial state, i.e. initial order of rows: $\pi_0 = \{c_1, c_2, \ldots, c_m\}$
$\boldsymbol{L}^{\pi} \leftarrow \boldsymbol{L}^{\pi 0}$	$\boldsymbol{L}^{\pi}{}_0$ – interval matrix $\boldsymbol{L}^{\pi}$ corresponding to the row permutation π_0
$t \leftarrow t_0$	t_0 – starting temperature: $t_0 = 0.8\,n$
$r_t \leftarrow r_0$ temperature $r_0 = m(m-1)/4$	r_0 – the number of iterations (moves) at starting

$Q^{\pi} \leftarrow Q^{\pi}_0$ Q^{π}_0 – the cost function that is the row permutation quality equaled to d^{π} –

(maximal number of 1's in a row of the interval matrix $\boldsymbol{L}^{\pi}$)

while $((\delta_t \geq 0.05\, r_t) \vee (t \neq 0))$ **do** not stopping criteria( )

begin

$i \leftarrow 0$ i is the serial number of iteration at a current temperature t

$\delta_t \leftarrow 0$ δ_t is the number of successful steps at a current temperature t

while $(i < r_t)$ **do** for $t = t_{const}$

begin

$\pi^* \leftarrow$ generate random neighbor of π

$\boldsymbol{B}^* \leftarrow \boldsymbol{B}^{\pi *}$ building Boolean matrix $\boldsymbol{B}$ for π^*

$\boldsymbol{L}^{\pi *} \leftarrow \boldsymbol{L}^{\pi}(\pi^*)$ building interval matrix for π^*

$Q^{\pi *} \leftarrow Q^{\pi}(\pi^*)$ evaluating $\boldsymbol{L}^{\pi *}$

$\triangle \leftarrow Q^{\pi *} - Q^{\pi}$

if $((\triangle \leq 0) \vee (\text{random}(0,1) < e^{-\triangle/t}))$ **then**

$\pi \leftarrow \pi^*$; $\boldsymbol{B} \leftarrow \boldsymbol{B}^{\pi *}$; $Q^{\pi} \leftarrow Q^{\pi *}$;

$\delta_t \leftarrow$ increase the number of successful steps(δ_t)

end while

$t \leftarrow$ reduce temperature(t); new temperature will be 0.85 of current value

end while the best solution is obtained

Figure 7. Simulated annealing folding procedure

Constrained multiple folding

The discussed procedure of multiple folding will be modified when some constraints on the form of folded structure are imposed. Such folding constraints can be divided on the following basic groups.

1. Structure-defined constraints. This class of constraints is defined by an array structure type in use. For example, use of PLA forces to take into consideration that its layout consists of AND-plane and OR-plane.

2. Input/Output constraints. This class of constraints depends on the environment in which the folded structure should be placed. For example, these can be the ordering of input/output directions, relative positions of input/output lines, grouping of input/output lines.

3. Electrical constraints. This class of constraints forces to limit the folding type. For example, it could be only simple or bipartite folding.

4. Physical Constraints. This class of constraints concerns to input lines carrying a signal and its inversion. For example, the appropriate array structure columns (or rows) may not share the line, so they cannot be folded, and sometimes they should be placed to each other as close as possible.

Further we show how we take proper account of some constraints imposed on folding demonstrating on the example of simple folding and PLA folding.

PLA folding via the method of simulated annealing

A PLA realizes a collection of Boolean functions in disjunctive normal form. It is a two-dimensional array consisting of an AND-plane and OR-plane. Its vertical lines are assigned with the input variables (and their complements) and output variables. Inputs run vertically through the AND-plane that generates signals on its rows, which are used as inputs to the OR-plane. The PLA folding allows some its columns to share a vertical line (or rows to share a horizontal line). An example of the PLA structure and its free folded form is shown in Figure 1. Here, a dot means placing a transistor on a cross point of vertical and horizontal lines.

PLA layout has structure-defined constraint that disallows a column of the AND-plane to share the same vertical line with a column of the OR-plane. Then, any physical realization of a folded PLA must ensure inputs/outputs be connected to the PLA outside. In [Wong, 1988] two possible architectures of implementation of routing lines carrying input/output signals to outside (with the help of horizontal auxiliary connection lines) and the modification of the simulated annealing process are shown.

So, PLA multiple folding is the constrained multiple folding of an array structure. It comes easily to take into account the first constraint when formulating the simulated annealing procedure. In this case the interval matrix $\boldsymbol{L}^{\pi}$ consist of two column submatrixes corresponding to the AND- and OR-planes, and we should distinguish between input $d^{\pi in}{}_i$ and output $d^{\pi out}{}_i$ densities of the matrix $\boldsymbol{L}^{\pi}$ rows. Then we construct two interval graphs $G^{in} = (V^{in}, E^{in})$ and $G^{out} = (V^{out}, E^{out})$ for the AND- and OR-planes. And the folding size associated with a row permutation π equals the sum of chromatic numbers of these graphs or it equals the maximal density d^{π} that is calculated as the sum of maximal input $d^{\pi in}$ and output $d^{\pi out}$ densities. Thus the simulated annealing procedure for solving the task of such constrained multiple folding distinguishes from the above formulated one only with the calculating maximal density d^{π} and accordingly the cost function Q^{π}.

In exactly the same way we can take into account the second constraint (concerning signal input and output), it is made by means of augmenting intervals in a proper way the structure have place to feed signals to its inputs and from its outputs. Thus we take a proper account of increasing interval lengths on the step of transforming matrix $\boldsymbol{B}^{\pi}$ into the interval matrix $\boldsymbol{L}^{\pi}$. The folded form of PLA from Figure 1,a that takes into account the routing lines is depicted in Figure 8.

Simple folding via the method of simulated annealing

Now we consider how to take into account more complex constraints – electrical constraints resulted to limit the folding type, for example, it could be only simple folding. Remember that folding is called simple if utmost two columns (rows) are allowed to share a single physical vertical (and/or horizontal) line.

Simple column folding has an evident advantage over multiple columns folding because external signals could connect to the folded structure either from the top, or the bottom of a folded structure because there is at most one break in any column. This simplifies routing signals, in addition, despite multiple folding might result in larger area reduction, simple folding allows reduce occupied routing area.

To take into account the peculiarity of simple folding in [Wong, 1988] it is proposed to use the same annealing schedule as for multiple folding but with other cost function (2). The function value depends on the matching number of undirected graph of pairwise compatibility (foldability) of PLA structure columns. Thus, at each simulated annealing iteration, we have to make the laborious procedure of compatibility graph construction and its matching number computation.

We propose to reduce the laboriousness of simple folding via simulated annealing method thanks to single implementation of the mentioned procedure of compatibility graph construction and its matching number computation. That cannot allow to obtain maximum folding in some cases but taking into account that simulated annealing method is a heuristic method that does not guarantee the optimum, the proposed idea is good enough.

The main point is based on the heuristics that we can get a good simple folding on basis of a good multiple folding. Thus, first we find out the best row permutation of array structure under folding using the proposed above procedure of multiple folding via the method of simulated annealing. Then we don't find array structure multiple folding itself but we will search for its best simple folding.

So, let we have now row permutation π and the corresponding interval matrix $\boldsymbol{L}^{\pi}$ (for the considered example it is shown in Figure 5). Let remember two intervals $\boldsymbol{l}_i$, $\boldsymbol{l}_j$ are compatible if they do not intersect, a pair c_i, c_j of corresponding columns are foldable and may be assigned the same vertical line in a proper layout. Let $H = (V, E)$ be an undirected graph, where the vertices of V correspond to intervals and $(v_i, v_j) \in E$ if and only if $\boldsymbol{l}_i$ and $\boldsymbol{l}_j$ don't intersect, ($\boldsymbol{l}_i \cap \boldsymbol{l}_j = \varnothing$). The size of simple folding associated with row permutation π equals to the matching number of the graph $H = (V, E)$ and the matching corresponds to collections of folding pairs.

A matching M in a graph H is a set of pairwise non-adjacent edges; that is, no two edges share a common vertex. Each edge in M defines a folding pair, and a set M specifies a folding set. Correspondingly, a maximum matching (that contains the largest possible number of edges) gives a folding set of maximum size. The number of columns of folded such a manner structure is equal the matching number plus the number of unmatched vertices of the graph H.

For our example the graph $H = (V, E)$ (corresponding to interval matrix $\boldsymbol{L}^{\pi}$ in Figure 5) is shown in Figure 9. Here all graph edges, but matching, are shown as thin lines and edges of the matching edges are shown as heavy lines. The folding variant of PLA shown in Figure 1,a is given in Figure 10.

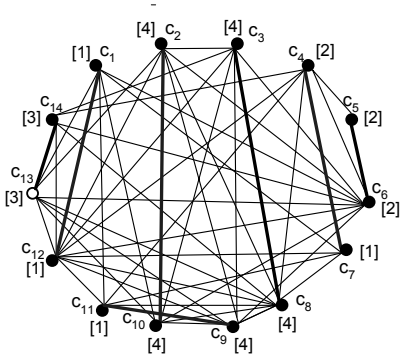

Figure 9. Compatibility graph $H = (V, E)$

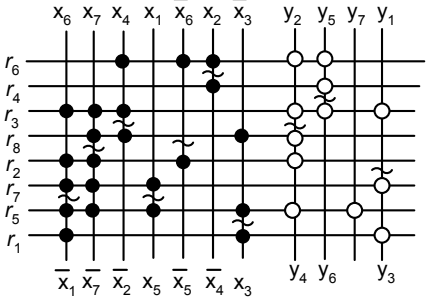

Figure 10. PLA simple folded form

Experimental results

The simulated annealing algorithms for multiple and simple folding (have been described above) were formulated for one of the types of regular structures, PLA structure, and were realized using Visual C++. Then they were compared on the stream of benchmarks [Berkeley, 2006].The objectives of performed computer experiments are to state:

1) the degree of PLA area reduction that could be achieved by both simple and multiple folding;

2) the dependence of degree of PLA area reduction on PLA AND- and OR-planes sparseness;

3) the comparative evaluations of two types PLA folding: simple and multiple ones.

As the basic parameter of PLA structures governed the degree of PLA compaction the density of PLA AND- and OR-planes was accepted. The density of PLA planes is the percent of active transistors in them (in relation to number of all transistors), or the PLA density is the ratio between the number of unit elements of Boolean matrix $\boldsymbol{B}$ (Figure 2) to whole number of its components.

The results of the testing are given in Figure 11; at the bottom of the Figure, in X-direction, bench PLA names are placed accompanied their densities. In Y-direction there are values of PLA area compaction expressed in percentage, that is $\frac{k}{n}\%$, where n is the number of PLA columns under folding and k is the number of the folded PLA columns.

Experimental results allow to draw a conclusion concerning the domains of applicability of simple and multiple folding. The results indicate that:

1) multiple folding gives better results for sparse PLA cases, or numerically in cases when PLA density is less than 20%;

2) simple folding gives better results for dense PLA cases, or numerically in cases when PLA density is more than 25%;

3) in the range of PLA densities 20 – 25% multiple and simple PLA foldings are competed with each other.

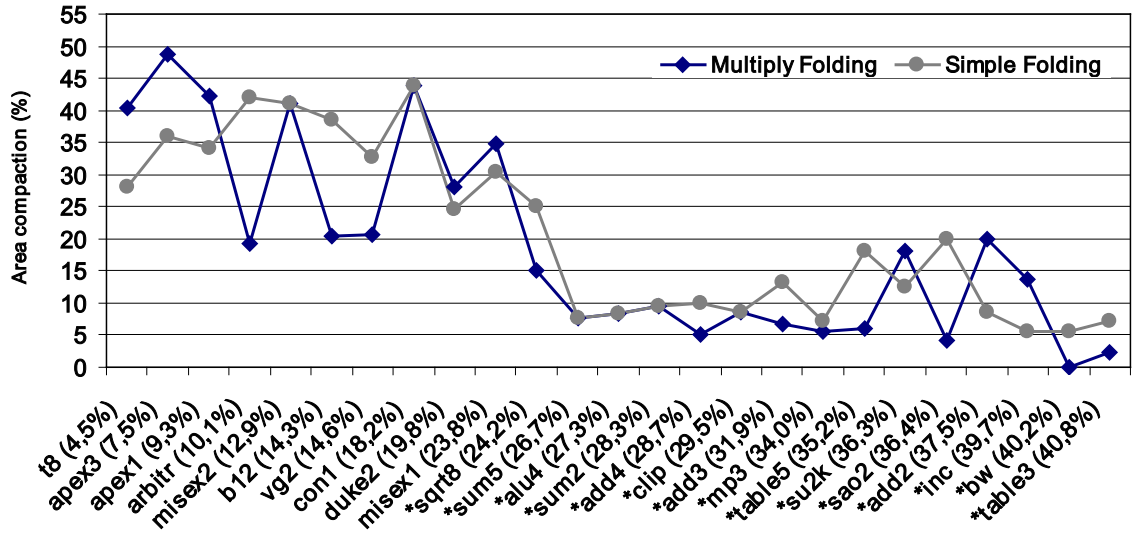

Figure 11. The results of experimental investigations of multiple and simple folding using simulated annealing algorithm

Conclusion

The problem of reducing the area of the layout of two dimensional array structures is investigated. We consider probabilistic heuristic algorithms based on simulated annealing procedure, they are well suited for compacting sparse structures having not great percentage of active elements (transistors).

Simulated annealing folding algorithms are investigated for the case of PLA multiple and simple column folding. The results of investigation show that 1) simulated annealing can give enough good results in the sense of area reduction, and 2) simple folding behaves not worse comparing with multiple folding on the stream of PLAs have been considered allowing for finding often solutions with the same cost function.

Bibliography

[Ullman, 1984] D. Jeffrey Ullman Computational aspects of VLSI. Rockville, Md.: Computer Science Press, 1984, 495 p.

[Hachtel, 1982] G.D. Hachtel, A.R. Newton and A.L. Sangiovanni-Vincentelli. An Algorithm for optimal PLA Folding. In: IEEE Trans. Computer-Aided Design of Integrated Circuit Syst., 1982, vol. CAD–1, No 2, pp. 63–77.

[DeMicheli, 1983] G. DeMicheli and A. Sangiovanni-Vincentelli. A. Multiple Constrained Folding of Programmable Logic Arrays: Theory and Applications. In: IEEE Trans. Computer-Aided Design, 1983, Vol. CAD-2, No 3, pp. 151–167.

[Wong, 1988] D.F. Wong, H.W. Leong, C.L. Liu. Simulated Annealing for VLSI Design. Kluwer Academic Publ., Boston, 1988, 220 p.

[Lee, 1995] Fu-H. A. Lee. Parallel simulated annealing on a message-passing multi-computer, Doctor of philosophy thesis, UTAH STATE UNIVERSITY, Logan, Utah, 1995.

[Kirkpatrick, 1983] S. Kirkpatrick, Jr. C. D. Gelatt and M. P. Vecchi. Optimization by simulated annealing. In: Science, 1983, 220 (4598), pp. 671–680.

[Van Laarhoven, 1987] P.J.M. Van Laarhoven and E. H. L. Aarts. *Simulated Annealing: Theory and Applications*. D. Reidel, Dordrecht, 1987.

[Hsu, 1999] W.-L. Hsu and T.-H. Ma. Fast and simple algorithms for recognizing chordal comparability graphs and interval graphs. *SIAM J. Comput.*, 1999, Vol. 28, No 3, pp.1004–1020.

[Golumbic, 1977] M.C. Golumbic. The complexity of comparability graph recognition and coloring. Computing, 1977, Vol. 18, pp. 199-208.

[Greening, 1995] D. R. Greening. Simulated Annealing with Errors, Doctor of philosophy thesis, UNIVERSITY OF CA, Los Angeles, 1995.

[Hashimoto, 71] A. Hashimoto, J. Stevens. Wire Routing by Optimizing Channel Assignment Within Larger Apertures. In: Proc. of 8th Design Automation Workshop, 1971.

[Berkeley, 2006] Berkeley PLA test set [Electronic resource]. – Mode of access: http:// www1.cs.columbia.edu/ ~cs4861/sis/ espresso-examples/. – Date of access: 03.05.2006.

The research was partially supported by the Fond of Fundamental Researches of Belarus (Project Ф10P–035).

Authors' Information

Liudmila Cheremisinova *– Principal Researcher, The United Institute of Informatics Problems of National Academy of Sciences of Belarus, Surganov str., 6, Minsk, 220012, Belarus, e-mail:* cld@newman.bas-net.by

Major Fields of Scientific Research: Logic Design, CAD systems, optimization

Irina Loginova *– Senior Researcher, The United Institute of Informatics Problems of National Academy of Sciences of Belarus, Surganov str., 6, Minsk, 220012, Belarus, e-mail:* cld@newman.bas-net.by

Major Fields of Scientific Research: Topological Design, CAD systems