On Generating Two Dimensional CMOS Cells

Jack A. Feldman[1] *and Israel Koren*[2]

1 IBM Israel Scientific Center, Technion City, Haifa 32000, ISRAEL
2 Dept. of Electrical and Computer Engineering, Univ. of Mass. Amherst MA 01003

Abstract. Most standard cell generators for CMOS circuits follow the single row (of transistor pairs) layout style. Multi-row layouts are then obtained by placing and interconnecting basic cells which were generated in the single row style.

In this paper we present an algorithm for generating two dimensional layouts. The number of rows (of transistor pairs) can be imposed by the user. This aspect ratio is selected so as to either minimize the total cell area or meet some height or width requirements.

We illustrate our algorithm through a set of examples, ranging from simple CMOS circuits to complex ones. We then compare the generated multi-row layouts to single row layouts of the same circuits. This comparison demonstrates the advantage of the two dimensional style over the single row style especially for more complex CMOS circuits.

1. INTRODUCTION

Layout generation of standard cells for *CMOS* circuits has been studied extensively. Most research focused on the layout style proposed by Uehara and vanCleemput [18], namely a linear arrangement (or a row) of transistor pairs. Each pair consists of a p –type transistor and a n –type transistor, which are vertically aligned at the center of their gates. Diffusion runs horizontally along each side of the row. Whenever adjacent diffusion ports belong to the same net they are connected as part of the diffusion run, otherwise a gap (or a break) has to be introduced in the diffusion run. Additional wiring is usually required to complete the internal connections. These wires are placed between and on top of the two diffusion runs.

The algorithms reported in [13], [15], [18] , and [19] search for the optimal solution in which as many consecutive transistor pairs as possible abut in their adjacent diffusion ports. In [8] and [14] the authors propose algorithms that can handle general circuits but are of heuristic nature, tuned towards wire density minimization. Finally, the algorithm suggested in [1], employs a depth first search procedure and prunes the search tree by using wire density constraints achieving an optimal solution.

The need for exploiting the second dimension lead to the development of strategies for generating layouts with multiple rows. Two approaches to lay out larger circuits in a multi-row style were proposed: 1) create a single row layout for the given large circuit which is then folded to generate a multi-row structure. 2) create the layout of (small) basic circuits in the single row layout style which are then placed and interconnected in a multi-row structure to generate the given large circuit.

In [9], the authors propose an automated system that generates a multi-row layout from the circuit description at the transistor level. Their experiments show that two or three times denser layouts can be achieved compared to multi-row layouts obtained by placing and interconnecting basic circuits laid out in the single row style. Their algorithm uses initial placement and iterative improvements followed by topological routing and compaction. In [10], [3], and [12] the authors propose an expert system approach to the generation of two dimensional layouts. There the initial placement and routing are improved by applying a set of appropriate rules. However, no algorithmic details are provided in these papers.

These observations lead us to develop a more general algorithm, which attempts to optimize both vertical and horizontal dimensions at the transistor level, rather then perform optimization in one dimension as usually done by the above mentioned algorithms. Our technique results in denser layouts compared to layouts generated in the single row style.

In Section 2 we describe the layout image. Section 3 describes the algorithms for the transistor placement, and Section 4 presents the routing steps. Finally, we present some results and discuss them in Section 5.

2. IMAGE DEFINITION

Our algorithm génerates automatically the symbolic layout of a CMOS cell starting from its topological description. The circuits are described in terms of their constituent transistors. For each transistor the type (*p* or *n*), the connections to its gate, drain and source ports and the device width and length are specified. The generic image of the layouts generated by our algorithm is illustrated in Figure 1. The layout has alternating *p* and *n* transistor rows arranged back to back. Power and ground buses are implemented in metal 1 layer and lie between consecutive transistor rows of type p and type n, respectively. The number of transistor rows versus transistor columns is a parameter of the algorithm. This allows to generate layouts with different aspect ratios, which is useful in the following situations:

- A layout with minimal total area is required (this occurs when constructing a library of building blocks).
- A layout that meets imposed dimensions, not necessary the minimum ones, is required (this occurs when considering the environment in which the cell is used).

Horizontal routing channels are inserted between consecutive p and n transistor rows, for connecting the transistor ports that border the channel. Vertical routing channels extend the left and right margins of the layout, and are used for connecting transistor ports that are located along different horizontal channels (see Figure 1).

Diffusion, polysilicon and metal 1 layers are used to accomplish internal routing. The inputs and outputs to the circuit are not restricted to specific locations in the layout. We assume that intercell connections are accomplished in additional metal layers which allow to reach *I/O* terminals on top of the underlying cells.

The placement grid is an $n \times m$ matrix G, where g_{ij} corresponds to a location where a transistor can be placed, n is the number of rows, and m the number of columns. All locations g_{ij}, $1 \leq j \leq m$ that compose a row i, are reserved to one transistor type. The sequence of rows in the placement grid corresponds to a back to back arrangement of p-n transistor rows. For example, if $i = 6$ then rows 1, 4, and 5 are reserved for $p -$ type transistors, and rows 2, 3 , and 6 are reserved for $n -$ type transistors. The number of rows and the type of transistors that can be placed in the first row are parameters of the placement program. These two parameters completely define the placement grid for a given circuit.

3. PLACEMENT

The transistor placement problem is solved in two steps. In the first, transistors are assigned to locations on the placement grid. In the second step the transistor orientation is determined. Partitioning the transistor placement into two well-defined steps replaces the large solution space which is $O(n! \times 2^n)$, where n is the number of transistors, by two smaller solution spaces, of size $O(n!)$ and $O(2^n)$ for the transistor assignment step and transistor orientation step, respectively.

3.1. TRANSISTOR ASSIGNMENT

The main goal when assigning transistors to locations in the placement grid is to encourage placements for which adjacent transistors share the same nets as much as possible. The definition of the objective functions and the solution technique used are discussed in the following paragraphs.

3.1.1. Objectives

A significant reduction in area results if transistors placed in consecutive locations $g_{i,j}$, $g_{i,j+1}$, in a row i, share the same net in their adjacent drain/source ports. The reduction due to drain/source abutting results from the following:

- No spacing has to be introduced between consecutive transistors.
- Connection between the adjacent drain/source ports is implemented directly in diffusion without any extra wires.
- One or two contacts are saved, if the corresponding net is a multi-port or a two port net, respectively.

Figure 2 illustrates two different permutations of three transistors, with abutting and non-abutting diffusion ports. In case (b), two spaces between the adjacent transistors, four contacts and two wires are saved compared to the configuration illustrated in Figure 2a.

Another factor that eases the routing and contributes to the area reduction is related to transistors placed in consecutive locations $g_{i,j}$, $g_{i+1,j}$ in the same column. Here a substantial area reduction is obtained if aligned consecutive transistors share the same net in their gates. In Figure 3a the connection between aligned gates is accomplished by a straight wire implemented in polysilicon, without any contacts. While, as shown in Figure 3b, if the gates to be connected are not aligned, the necessary connections must be accomplished in the channel introduced between the adjacent transistor rows. In summary, the objectives considered for the assignment algorithm are:

1. In the horizontal direction - maximize the number of diffusion abutments.
2. In the vertical direction - maximize the number of polysilicon abutments.

3.1.2. Mathematical Formulation

The problem of assigning transistors to grid locations can be formulated as the known quadratic assignment problem, which is stated as follows: given a set $E = (1,...,e)$ of elements and a set $L = (1,...,l)$ of locations, find a one to one mapping from set E into set L where $l \geq e$. Let:

$Q = [q_{ij}]$ be a $e \times e$ matrix, and
$D = [d_{ij}]$ be a $l \times l$ matrix.

where:

$q_{ij} =$ is the measure of the affinity (or attraction) between element i and element j,
$d_{ij} =$ is the distance between location i and location j.

Let S be the set of all possible mappings of set E into set L. Let $\rho \in S$, be a particular mapping, and let $\rho(i)$ denote the location to which element i was assigned in the mapping ρ. Consider now the functional:

$$H(\rho) = \sum_{ij} q_{ij} d_{\rho(i)\rho(j)} \tag{1}$$

Then, the quadratic assignment problem can be stated as follows:

$$\min_{\rho \in S} H(\rho) \tag{2}$$

3.1.3. An Interpretation for the Affinity and Distance Matrices

The transistors that constitute a given circuit compose the set E , and the $n \times m$ locations compose the set L. Then, the affinity matrix reflects the connectivity between the various transistors, and the distance matrix provides some metric on the placement grid.

The first objective considered for the transistor assignment problem is to maximize the number of horizontal diffusion abutments. We denote here the affinity matrix by q^h_{ij} and the distance matrix by d^h_{ij}. Let T_i and T_j denote two different transistors, then the element q^h_{ij} is set to,

$$q^h_{ij} = \begin{cases} 2 & \textit{if } T_i \text{ and } T_j \text{ have both diffusion ports labelled by the same nets} \\ 1 & \textit{if } T_i \text{ and } T_j \text{ have one diffusion port labelled by the same net} \\ 0 & \textit{otherwise} \end{cases} \tag{3}$$

We label the locations in the placement grid from 1 to $n \times m$, with the convention that the rows of the matrix are scanned from left to right and from top to bottom. Let x_k and y_k

denote the $x-$ and $y-$coordinates associated with location k $(1 \leq k \leq n \times m)$ in the placement grid. Then define:

$$d_{ij}^{h} = \begin{cases} 0 & \text{if } y_i = y_j \wedge |x_i - x_j| \leq 1 \\ c & \text{otherwise} \end{cases} \tag{4}$$

According to these definitions, if T_i and T_j have one or two common diffusion ports and are assigned to adjacent locations in the same row of the placement grid, then no penalty is added to the cost function $H(\rho)$. For any other assignment a penalty equal to c (or $2c$ in the case where both diffusion ports of T_i and T_j belong to the same nets) is added to $H(\rho)$. We distinguish between the case where T_i and T_j share one diffusion port or two, since for the latter one two diffusion abutments are possible. If transistors T_i and T_j have no common diffusion port, no penalty is added to the cost function $H(\rho)$ independent of their positions.

For the second objective considered for the transistor assignment, namely, maximizing the number of vertical polysilicon abutments, we define the affinity matrix q_{ij}^{v} and the distance matrix d_{ij}^{v}, as follows:

$$q_{ij}^{v} = \begin{cases} 1 & \text{if } T_i \text{ and } T_j \text{ share the same net in their gates} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$d_{ij}^{v} = \begin{cases} 0 & \text{if } x_i = x_j \wedge |y_i - y_j| \leq 1 \\ c & \text{otherwise} \end{cases} \tag{6}$$

In this case, if transistors T_i and T_j share the same net in their gates and were assigned to consecutive locations in the same column of the placement grid, then no penalty is added to the cost function $H(\rho)$; for any other assignment a penalty equal to c is added to $H(\rho)$. If transistors T_i and T_j do not share the same net in their gate no penalty is added to the cost function $H(\rho)$, independent of their location.

3.1.4. Solution Technique

The quadratic assignment problem has proven to be NP-complete [5] and consequently, methods for finding the optimal solution are computationally feasible only for small problems ($n \leq 15$). These include the branch and bound methods proposed by Lawler [11] and Gilmore [7], and the cutting plane method proposed by Bazara and Sherali [2] . Since in practice we deal with much larger problems, various heuristic algorithms have been proposed to solve the quadratic assignment problem. A very efficient constructive algorithm that belongs to this class is due to Graves and Whinston [6].

Define the *k*th *partial permutation* as a mapping of the set E into the set L that leaves the first k element-location pairs unchanged. The selection criterion is based on the mean value over all possible completions of a $k-$partial permutation. An expression for this mean value has been derived in [6] and is reproduced in equation 7.

$$\mu = \sum_{i \in F} \sum_{j \in F} q_{i,j} d_{\rho(i),\rho(j)} + \frac{1}{n-k} \sum_{i \in F} \sum_{j \in \tilde{F}} \sum_{t \in \tilde{T}} (q_{i,j} d_{\rho(i),t} + q_{j,i} d_{t,\rho(i)}) + \frac{1}{(n-k)(n-k-1)} \sum_{u \in \tilde{F}} \sum_{j \in \tilde{F}} q_{j,u} \sum_{s \in \tilde{T}} \sum_{t \in \tilde{T}} d_{s,t} \tag{7}$$

where:

$F = \{i_1,\dots,i_k\}$, the set of assigned elements.
$\tilde{F} = \{i_{k+1},\dots,i_n\}$, the set of unassigned elements.
$T = \{j_1,\dots j_k\}$, the set of occupied locations.
$\tilde{T} = \{j_{k+1},\dots j_n\}$, the set of free locations.

An important property of the scheme in [6] is that the enumeration of all the possible assignments is not required. For each feasible unassigned element-location pair (out of $(n-k)^2$ combinations) we compute its mean value, and select that element-location pair for which the smallest mean is obtained. Note that expression (7) holds valid for any affinity and distance matrices. Let μ_h and μ_v denote the mean value for the horizontal and vertical objective, respectively. Then the expression for μ_h is obtained by substituting q_{ij} by q_{ij}^h and d_{ij} by d_{ij}^h in (7). Similarly, we substitute q_{ij} by q_{ij}^v and d_{ij} by d_{ij}^v in (7) to obtain μ_v . In order to combine the two objectives, the two mean values μ_h and μ_v must be normalized. The normalization factors correspond to the mean values in the case where all elements are unassigned and all locations are free, i.e. $F = T$, and $\tilde{F} = \{i_1,\dots,i_n\}$, and $\tilde{T} = \{j_1,\dots j_n\}$. The normalization factors are given therefore, by:

$$N_h = \frac{1}{n^2} \sum_{u \in \tilde{F}} \sum_{j \in \tilde{F}} q_{ju}^h \sum_{s \in \tilde{T}} \sum_{t \in \tilde{T}} d_{st}^h \tag{8}$$

$$N_v = \frac{1}{n^2} \sum_{u \in \tilde{F}} \sum_{j \in \tilde{F}} q_{ju}^v \sum_{s \in \tilde{T}} \sum_{t \in \tilde{T}} d_{st}^v \tag{9}$$

Finally, we combine the two normalized objectives and attempt to minimize a single expression, i.e.,

$$\mathrm{M} = w_h \frac{1}{N_h} \mu_h + w_v \frac{1}{N_v} \mu_v \tag{10}$$

where w_h and w_v are weights assigned to the two objectives to allow us to control their relative significance.

3.1.5. Enumeration Scheme

The enumeration scheme employed, constructs one path within a virtual search tree of order $O(n!)$. In step k a new element-location pair is selected out of the remaining $(n-k)$ unassigned elements and $(n-k)$ unoccupied locations. For each feasible element-location pair out of $(n-k)^2$ possibilities, we compute the corresponding mean value and select the one having the smallest mean. An element-location pair is feasible if the transistor and the location on the placement grid are of the same type.

3.1.6. Complexity of the Algorithm

A straightforward implementation of the above search algorithm has an $O(n^6)$ complexity. It results from the fact that n steps are required to map the n elements into the n locations, where at step k there are $(n-k)^2$ combinations of unassigned elements and unoccupied locations. Finally, for each element-location pair considered, the mean value has to be computed and this has an $O(n^3)$ complexity. However, a careful implementation of the mean value calculation can save $O(n)$ computations. This is due to the following observation: the mean value in step $k+1$ differs from the mean value in step k only by the contribution of the currently selected element-location pair. Accordingly, after completion of step k we compute and store the mean value of the kth partial permutation. In step $k+1$ we compute the contribution of the currently selected element-location pair and add it to the mean value of the kth partial permutation. Let w and v denote the currently selected element-location pair, where $w \in \tilde{F}$ and $v \in \tilde{T}$. Then, the contribution of the pair w,v to the mean value of the $(k+1)$th partial permutation is given by the following expression:

$$
\begin{aligned}
C = {} & \sum_i (q_{iw} d_{p(i)v} + q_{wi} d_{vp(i)}) + \sum_s \sum_l (q_{sw} d_{lv} + q_{ws} d_{vl}) \\
& - \left(\sum_i \sum_t (q_{iw} d_{p(i)t} + q_{wi} d_{tp(i)}) + \sum_i \sum_j (q_{ij} d_{p(i)v} + q_{ji} d_{vp(i)}) \right) \\
& + \sum_i (q_{iw} d_{p(i)v} + q_{wi} d_{vp(i)}) - \sum_s (q_{ws} + q_{sw}) - \sum_l (d_{vl} + d_{lv})
\end{aligned}
\tag{11}
$$

where $i \in F^k$, $j \in \tilde{T}^k$, $t \in \tilde{F}^k$, $s \in \tilde{F}^{k+1}$, $l \in \tilde{T}^{k+1}$.

Emplying (11) the computation of the mean value requires $O(n^2)$ steps as opposed to (7) which requires $O(n^3)$ steps and consequently, $O(n^5)$ is the complexity of our search algorithm.

3.1.7. Strategy for Determining the Weights

Weights w_1, w_2 are attached to the above two objectives. Intuitively it is clear that the ratio between w_1 and w_2 for which the best result is obtained, depends on the circuit topology. An additional optimization for the best w_1/w_2 ratio is possible but it does not justify the effort. However, the following heuristic strategy can be employed to select values for w_1 and w_2:

- Set $w_1 = 1$ and $w_2 = 0$ and find the number x_1 of potential diffusion abutments for the given circuit.
- Set $w_1 = 0$ and $w_2 = 1$ and find the number x_2 of aligned gates that share the same net.
- Set $w_1 = 1$ and $w_2 = 1$ and find x'_1 and x'_2, the number of potential diffusion abutments and the number of aligned gates that share the same net, respectively.

The values $(x_1 - x'_1)$ and $(x_2 - x'_2)$ can be used as a guideline to determine which one of the two objectives should be assigned a higher weight.

3.2. OPTIMAL ORIENTATION OF TRANSISTORS

In the first phase of the placement procedure, transistors were assigned to locations. However, no decision concerning the transistor orientation was made. For each transistor two orientations are possible, the drain port can be at the left side and the source port at the right side or vice versa. In what follows we present the objectives considered for transistor orientation and then proceed with the proposed solution technique.

3.2.1. Objectives

Adjacent diffusion ports in the same row which share the same net are preferred over diffusion breaks. This is the primary objective considered for the transistor orientation step. The secondary objective considered is related to consecutive transistors in the same column. Here an orientation of consecutive transistors in the same column such that their aligned diffusion ports share the same net, as shown in Figure 4a., is preferred over the orientation shown in Figure 4b. The transistor orientation in case (a) is better since the connection is implemented by a straight wire and requires less contacts. Let $T^l_{(i,j)}$ be a transistor that was placed in row i column j and has orientation l. Let e^h denote the horizontal abutment cost for two transistors placed in consecutive columns in the same row. The horizontal abutment cost is defined as follows:

$$e^h(T^l_{(i,j)}, T^k_{(i,j-1)}) = \begin{cases} 1 & \textit{if } T^l_{(i,j)} \text{ and } T^k_{(i,j-1)} \text{ share the same net} \\ & \text{in their adjacent diffusion ports} \\ 0 & \textit{otherwise} \end{cases}$$

Let e^v denote the vertical alignment cost for two transistors placed in consecutive rows in the same column. The vertical alignment cost is defined as follows:

$$e^v(T^l_{(i,j)}, T^k_{(i-1,j)}) = \begin{cases} 2 & \textit{if } T^l_{(i,j)} \text{ and } T^k_{(i-1,j)} \text{ share the same net} \\ & \text{in their aligned diffusion ports} \\ 1 & \textit{if } T^l_{(i,j)} \text{ and } T^k_{(i-1,j)} \text{ share the same net} \\ & \text{in one of the aligned diffusion ports} \\ 0 & \textit{otherwise} \end{cases}$$

3.2.2. Solution Technique

The transistor orientation problem has been previously mentioned in works related to automatic cell generation in the single row style of p and n type transistor pairs. In [8] a branch and bound algorithm, that implicitly enumerates the entire solution space, was proposed. This solution technique is computationally too expensive. In [1], a dynamic programming algorithm was suggested for finding the optimal solution. This solution technique is extended in what follows to satisfy our requirements.

The dynamic programming technique is applicable due to the following two properties of this problem:

- Property 1: The optimal orientation of the j-th transistor column depends only on the $(j-1)$-th transistor column.
- Property 2: Both cost measures are additive functions.

The algorithm scans the $[n \times m]$ placement matrix, where n represents the number of rows and m represents the number of columns, from left to right, column by column. There are 2^n possible configurations for transistor orientations in a column. Let state (j,l) , where $1 \leq j \leq m$ and $1 \leq l \leq 2^n$, denote the orientation for transistors in column j according to configuration l. A state is coded by a sequence of n 0/1 bits where each bit stands for a transistor that belongs to column j, and its value reflects the transistor's orientation. For each state (j,l) two costs are computed. The first one reflects the best horizontal diffusion abutment cost for concatenating the transistors in columns j and $j-1$. Its expression is given by:

$$C^h_{(j,l)} = \max_{1 \leq k \leq 2^n} \{ C^h_{(j-1,k)} + \sum_{i=1}^{n} e^h(T^l_{(i,j)}, T^k_{(i,j-1)}) \} \tag{12}$$

where $C^h_{(j-1,k)}$ is the best horizontal diffusion abutment cost so far for state $(j-1,k)$.

The second one reflects the best vertical diffusion alignment cost for state (j,l). Its expression is given by:

$$C^v_{(j,l)} = \sum_{i=2}^{n} e^v(T^l_{(i,j)}, T^l_{(i-1,j)}) + \max_{1 \leq k \leq 2^n} \{ C^v_{(j-1,k)} \} \tag{13}$$

where $C^v_{(j-1,k)}$ is the best vertical diffusion alignment cost so far for state $(j-1,k)$.

Let the predecessor to state (j,k), denoted by $pred(j,k)$, be k_0, where k_0 is the index which maximizes expression (12). In case this index is not unique, we choose among all the indices k_0 the one which maximizes the expression in (13). Initially we have the states $(1,l)$, $l = 1..2^n$ with costs $C^h_{1,l}$ and $C^v_{1,l}$. For each state (j,l) we keep its horizontal and vertical costs $C^h_{j,l}$ and $C^v_{j,l}$ (which are computed by equation (12) and (13) above) and its predecessor $pred(j,l)$. For $j = m$ we choose among the states (m,l) the ones for which $C^h_{m,l}$ is maximum. If there is more then one such state, we break ties by $C^v_{m,l}$. The final solution is obtained by backtracking from $j = m$ to $j = 1$ using the predecessor states.

4. ROUTING

Once the transistor placement and orientation steps have been completed, we need to interconnect the various transistor ports. These wires will be routed through channels as illustrated in Figure 1. The horizontal channels are used to interconnect ports which are internal to these channels. Vertical channels are used to interconnect nets that lie in more than one horizontal channel. The main difference between these two channel types is that the ports in the horizontal channels are distributed along the four sides, whereas for the vertical channels, ports are distributed along one side only.

Polysilicon layer, and metal 1 layer are used to complete the interconnections in the horizontal and vertical channels. Polysilicon wires run only vertically while metal 1 wires run horizontally, as it is usually done in channel routing. Adjacent drain/source ports that belong to the same net are connected through diffusion.

The overall strategy used for the routing algorithm is a left to right scan of the routing template. The routing process is partitioned into five steps which are executed sequentially. These steps are as follows:

1. Determine and mark the nets that can be routed between consecutive transistor rows of the same type. The connections allowed between transistor rows of the same type are: straight vertical polysilicon wires that connect aligned transistor gates which belong to the same net and straight vertical diffusion wires that connect aligned diffusion ports which belong to the same net. This step attempts to avoid the routing of nets in the vertical channels, and as a result reduces the wire length, number of contacts and the wiring density in the routing channels.
2. Determine the nets to be routed in the left and right vertical channel. These are the nets that have their ports distributed on different horizontal channels, and were not connected in the previous step. Either the right channel or the left one is selected for each net so that wire length is minimized.
3. Routing of the left vertical channel. Assignment of wires to tracks is accomplished by interval graph coloring [16].
4. Routing of the horizontal channels. The algorithm used to decide upon the routing pattern is a variation of the greedy channel router [17].
5. Finally, routing of the right vertical channel is performed. Interval graph coloring is used again to assign wires to tracks in the right vertical channel.

5. RESULTS AND DISCUSSION

In this section we demonstrate the advantages that result from generating two dimensional layouts through a sample of circuits. These include: a full adder, multiplexor, decoder, 9 way AND gate, and 9 way OR gate. A layout for the full adder is illustrated in Figure 5. Table 1 illustrates the results obtained for the Full-Adder. It demonstrates the capability of generating layouts with different aspect ratios.

As was intuitively expected, circuits behave differently for various aspect ratios and the minimal area layout does not always occur for the same aspect ratio. Consequently, it is important to have the capability to generate layouts with different aspect ratios. This capability is also important if area constraints are imposed by the surrounding environment in which cells are used. In this case layouts must meet the imposed dimensions according to the available space on the chip. Note that for this example the minimal area layout was obtained when 6 rows were used. By varying w_1/w_2 the layout area might be further improved.

In order to measure the quality of the generated layouts they were compared in terms of area, with layouts generated automatically by a similar system developed at IBM [1]. The latter was developed to handle standard cells in the style of one dimensional arrangement of $p-n$ transistor pairs. There, the placement algorithm is based on a branch and bound technique that searches for the optimal solution in terms of diffusion breaks and wiring density. The same compactor was used as the back end tool for both systems. Thus, layouts were generated for the same technology with the same ground rules, which makes the comparison fair from this point of view. Table 2 presents the comparison of the results achieved by the two tools.

The results demonstrate that the layouts generated by our tool are competitive with the results obtained by the IBM tool. For larger and more complex circuits (e.g., full adder, multiplexor and decoder) better layouts (in terms of area) were generated by our tool, whereas for smaller circuits (e.g., AND and OR gates) the IBM tool performed better. As a general remark, as long as the wiring density for the linear transistor arrangement is kept under a certain value (say 5 wiring tracks) this layout style is the most effective one for packing the circuits. The two dimensional layout style becomes more effective in terms of the total area for larger circuits.

However, the flexibility for generating layouts with different aspect ratios for a given circuit, is the most important capability, since it allows to select that layout style which best fits the imposed requirements by the surrounding environment. It is our belief that CAD tools which allow dynamic exchange of layout styles in the process of composing the chip, will represent the future trend.

Our system was implemented in Pascal containing about 4000 lines, and has been executed on a 4381 IBM computer. Table 3 shows the computation time required for these circuits.

6. CONCLUSIONS

A new algorithmic method for laying out circuits from their schematic description, in the style of two dimensional layouts has been developed. The proposed image, has n-type and p-type transistor rows, where alternating $p-n$ transistor rows are arranged back to back. The presented algorithm does not impose structural limitations on the circuit topology, and accommodates well-defined optimization criterions.

One of the most important capabilities of the system is that it can deal with different aspect ratios for the generated layouts. This allows to use the layouts in a static context, where minimal area layouts are required, and in a dynamic context when considering the surrounding environment in which the layouts are used.

The presented results demonstrate that optimization in two dimensions can be efficiently exploited for reducing the total layout area.

Acknowledgements. This paper is based on the first author's M.Sc. research which was done in the department of Electrical Engineering at the Technion, Israel. The authors wish to thank Alfred Bruckstein for helpful discussions, and Shmuel Wimer and Ron Pinter for their useful comments.

REFERENCES

[1] Bar-Yehuda R, J.A. Feldman, R.Y. Pinter and S. Wimer, **Depth First Search and Dynamic Programming Algorithms for Efficient CMOS Cell Generation,** *Advanced research in VLSI, Proceedings of the Fifth MIT Conference,* 1988, 215-228.

[2] Bazaraa M.S., and H.D. Sherali **On the use of exact and heuristic cutting plane methods for the quadratic assignment problem.,** *J. Op. Res. Soc. 33,* 1980, 991-1003.

[3] Cesear T., E. Iodice, and C. Tsareff **PAMS: An expert system for parameterized module synthesis** *24th DAC,* 1987, 666-671.

[4] Fiebrich R.-D., Y.-Z. Liao, G. Koppelman, and E. N. Adams, **PSI - a Symbolic Layout System,** *IBM Journal of Research and Development 28, 5* (September 1984), 572-580.

[5] Garey M.R., and D.S. Johnson **Computers and Intractability: A Guide to the Theory of NP-Completeness.,** *W.H. Freeman,* 1979.

[6] Graves G.W., and A. Whinston **An algorithm for the quadratic assignment problem.,** *Mgmt Sci. 16,* 1970, 453-471.

[7] Gilmore P.C. **Optimal and suboptimal algorithms for the quadratic assignment problem.,** *J. SIAM 10,* 1962, 305-313.

[8] Hill D., **Sc2 - a Hybrid Automatic Layout System,** *Proceedings of ICCAD,* November 1985, 172-174.

[9] Hughes T.A., R. Salama and W. Liu, **BBC: A Module Generator for Back-to-Back Cells.,** *Proceedings of ICCAD,* 1986, 440-443.

[10] Kollaritsch P.W. and N.H.E. Weste, **TOPOLOGIZER: An expert system translator of transistor connectivity to symbolic cell layout** *IEEE J. of Solid-State Circuits,* 1985, 799-804.

[11] Lawler E.L. **The Quadratic Assignment Problem.,** *Mgmt Sci. 9,* 1963, 586-599.

[12] Lin S., and D. Gajski, **LES: A Layout Expert System,** *24th. DAC,* 1987, 672-678.

[13] Muller R., and T. Lengauer, **Linear Algorithms for two CMOS Layout Problems,** *Proc. Aegean Workshop Comput.,* July 1986.

[14] Miyashita H., T. Adachi, and K. Ueda, **An automatic cell pattern generation system for CMOS transistor-pair array LSI,** *Integration, Vol. 4,* 1986, 115-133.

[15] Nair, R., A. Bruss, and J. Reif **Linear Time Algorithms for Optimal CMOS Layout,** *VLSI: Algorithms and Architectures, P. Bertolazzi and F. Luccio (Eds), Elsevier (North-Holland),* 1985, 327-338.

[16] Ohtsuki T., **Layout design and verification,** *North-Holland,* 1986.

[17] Rivest, R. L., and C. M. Fiduccia, **A "Greedy" Channel Router,** *Proceedings of the Nineteenth Design Automation Conference,* June 1982, 418-424.

[18] Uehara T., and W. M. vanCleemput, **Optimal Layout of CMOS Functional Arrays,** *IEEE Transactions on Computers C-30, 5,* May 1981, 305-312.

[19] Wimer S., R.Y. Pinter, J.A. Feldman **Optimal Chaining of CMOS Transistors in a Functional Cell,** *IEEE Transaction on Computer Aided Design, Vol. CAD-6, No. 5,* 1987, 795-801.

No. of rows	w1	w2	Layout width	Layout length	Total area	No. of Poly adjacencies	No. of Diffusion abutments
2	1	1	143.7	72.9	10475.7	16	34
4	1	1	83	145.8	12101	23	30
4	1	0	101	143.8	14523.8	23	24
4	0	1	85.3	155.5	13264.1	8	30
4	1	2	71.9	132.8	9533.9	18	39
6	1	1	65.3	141	9207.3	26	38
6	1	0	92.6	189	17501.4	27	22
6	0	1	68.9	188.4	12980.7	11	37
8	1	1	67	214.9	14405	25	30
8	0	1	61	238	14518	11	34

Table 1. Results for FULL-ADDER

cell	total area single-row	total area multi-row	no. of rows multi-row
Full-Adder	9790.4	9207.3	6
Multiplexor	39427.3	37959	4
Decoder	44262.8	41760	4
And gate	6554.7	8477	4
Or gate	5216	7810	4

Table 2. Comparison table

Cell	No. of rows	No. of elements	Assignment CPU time (sec)	Orientation CPU time (sec)	Routing CPU time (sec)
Full adder	8	32	27	13	0.7
Multiplexor	8	62	245	31.6	1.5
Decoder	8	88	773	42.7	2.8
9 way AND gate	8	30	28	13.2	0.6
9 way OR gate	8	32	27	13	0.6

Table 3. Computation time

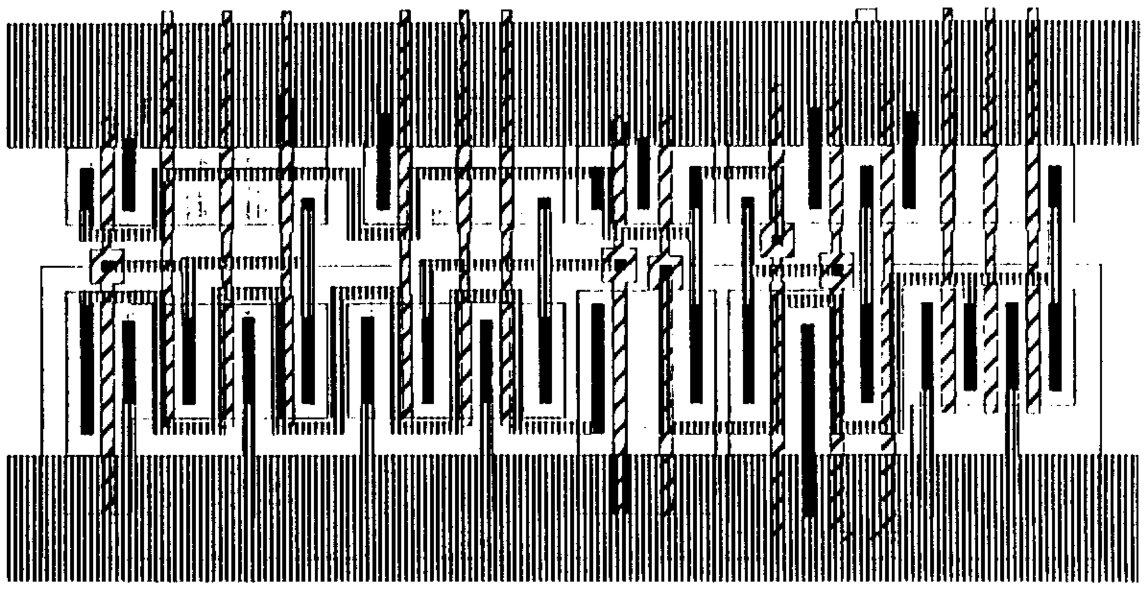

Figure 1. layout In the single row style.

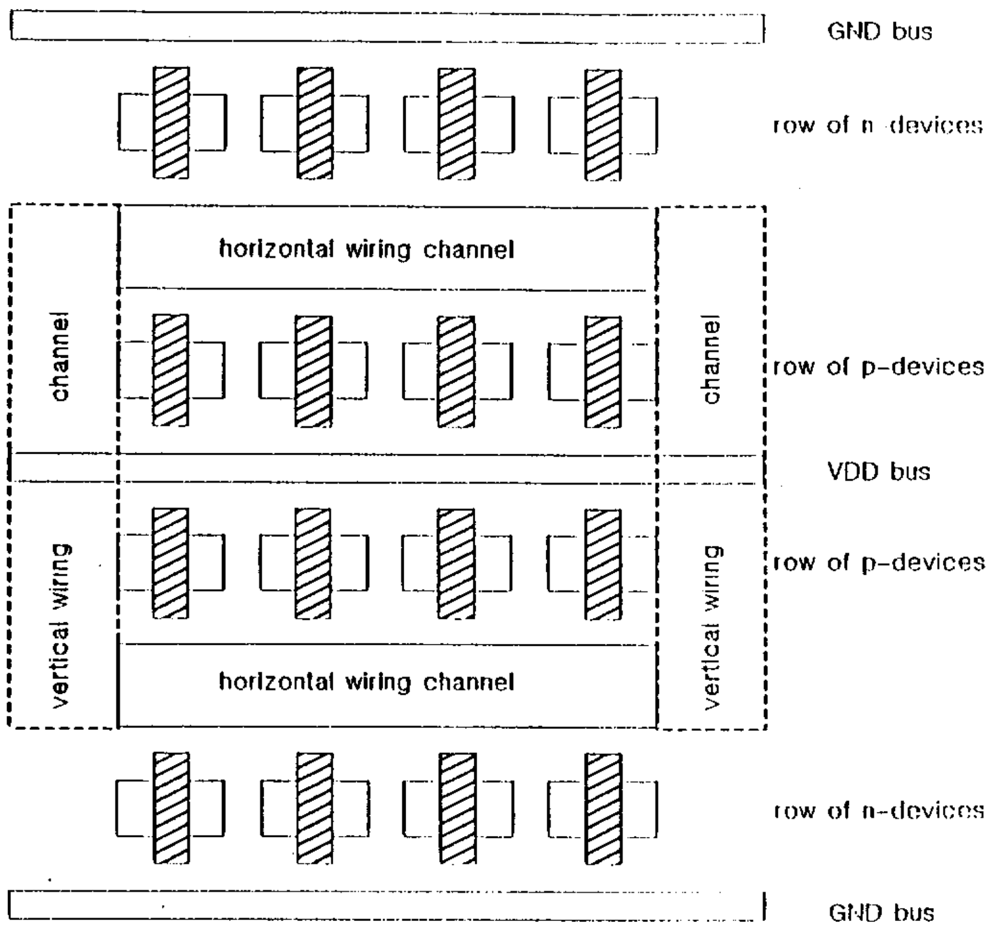

Figure 2. layout style.

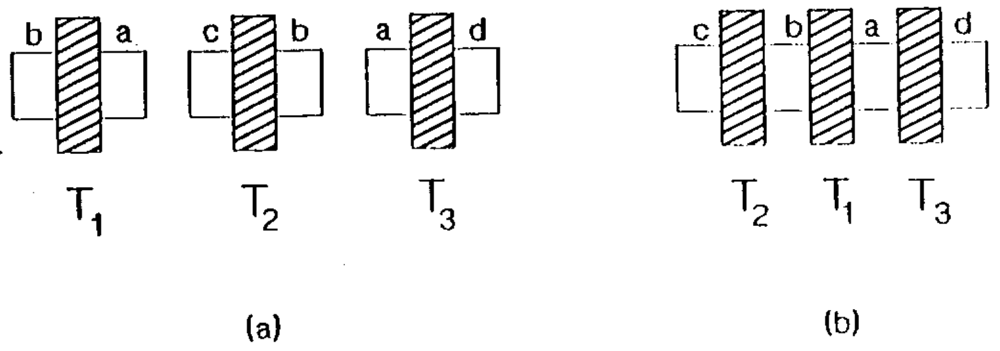

Figure 4. diffusion abutment versus non abutment.

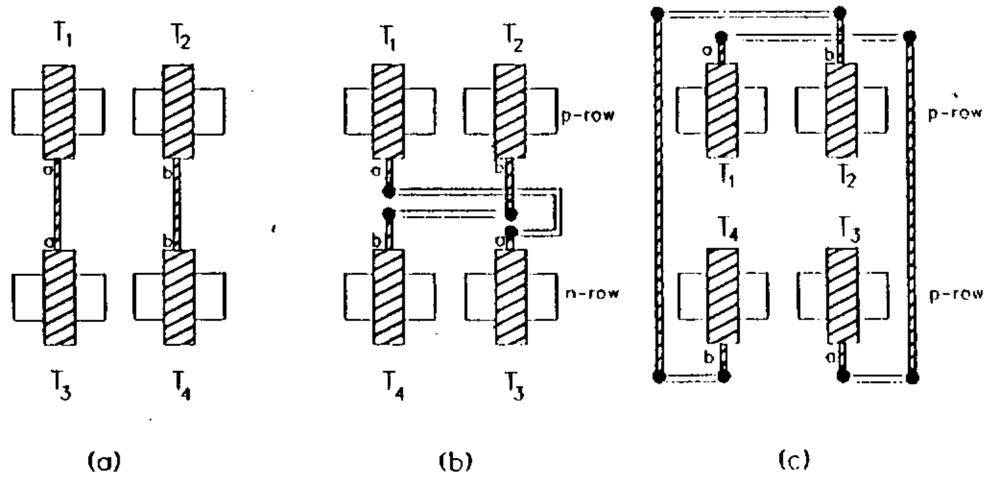

Figure 5. aligned transistor gates sharing the same net versus aligned transistors gates that belong to different nets.

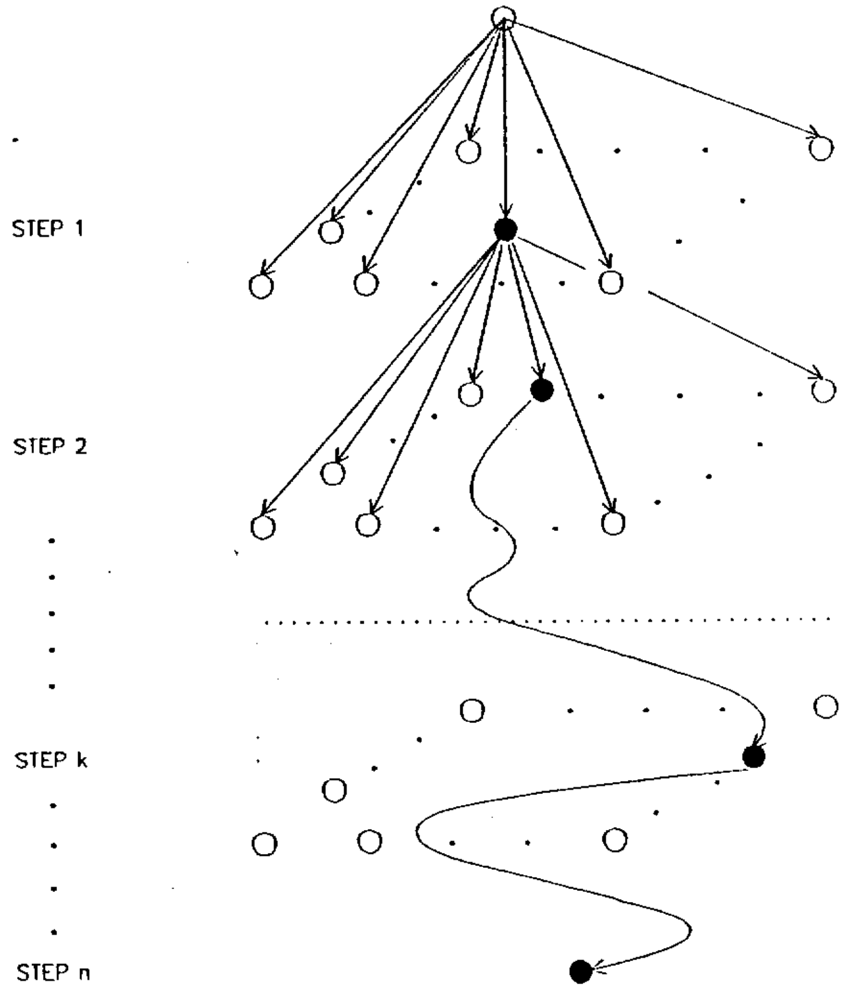

Figure 6. virtual search tree employed for the enumeration scheme.