Asian Journal of Information Technology, 5 (2) : 183-198, 2006

Vlsi Cells Placement Algorithm, Using The Kohonen Networks

[1]Hacène Azizi, [1]Khier Benmahammed and [2]Oum-el-kheir Aktouf
[1]Department of Physics, Faculty of Science, Sétif University, 19000 Algeria
[2]Inpg Grenoble France

Abstract: In this study, we study the relative placement of the cells at the integrated circuit drawing time. Studies showed that a particular type of artificial neural network, the KOHONEN network, can be used to position cells of identical size and form on a plan of mass without rails of connections. We presented a first algorithm, with simplifying assumptions, which did not give satisfactory results that pushed us to improve it to become a very powerful algorithm. Then, we approached the phase of the extrapolation of the algorithm, where the cells were of different size. We introduced the concept of terminal where each cell is represented by a set of terminals and where connections were done between these same terminals.

Key words: Artificial neural networks, KOHONEN networks, VLSI cells placement

INTRODUCTION

The artificial neural networks have been studied for several years. Their effectiveness makes it possible to hope for high performances. The privileged fields of these techniques remain the recognition and classification. Different applications of optimization are also studied using the artificial neural networks. They make it possible to apply heuristic distributed algorithms.

In this study, a solution to the problem of various cells placement at the time of the realization of an integrated circuit is proposed using the KOHONEN network.

The placement problem in VLSI design is the first phase in the process of designing the physical layout of a chip. This makes the placement problem of paramount importance, since the quality of the attainable routing is to a high degree determined by the placement.

PROBLEM OF FUNDAMENTAL PLACEMENT

The problem of the cells placement is posed when designing an integrated circuit. Once the functions of the circuit are defined, we'll be able to deduce an electric scheme fulfilling the desired function. The circuit is composed of a set of inter-connected functional units[3].

These units are even composed of basic elements like transistors or lines of connections; they are drawn and optimized individually. Thus they become basic units which we will call cells. When, the integrated circuit is built by connecting the cells, the problem of the placement is to define how the cells will be placed to maximize the performances of this circuit. It is a question of minimizing necessary surface.

The algorithms and the known methods to solve this problem suffer of the increasing in the computing time according to the number of the cells. To resolve this problem, we must impose constraints for the solution search.

In this part, we will analyze an elementary case of the placement problem basing us on the paper of HEMANI and POSTULA[2]. Thus, we admit the following simplifying assumptions

- All the cells are of cut and identical form.
- It doesn't exist rails of connection: the intercellular links are made by using open space between the cells.
- The cells are laid out on a unit grid of step.
- The minimized cost function takes into account only the length of connections.

The surface reserved for connections will be thus also minimized. This should have a beneficial effect on the total surface of the circuit.

A circuit made up of 9 cells will take for example the form described in the Fig. 1.

Table 1: The connections matrix of the circuit of Fig. 1

	0	1	2	3	4	5	6	7	8
0	0	5	0	5	2	0	0	0	0
1	5	0	5	0	5	0	0	0	0
2	0	5	0	0	2	5	0	0	0
3	5	0	0	0	5	0	5	0	0
4	2	5	2	5	0	5	2	5	2
5	0	0	5	0	5	0	0	0	5
6	0	0	0	5	2	0	0	5	0
7	0	0	0	0	5	0	5	0	5
8	0	0	0	0	2	5	0	5	0

Corresponding author: H. Azizi, Faculty of Science, Physic Department, Sétif University, 19000 Algeria

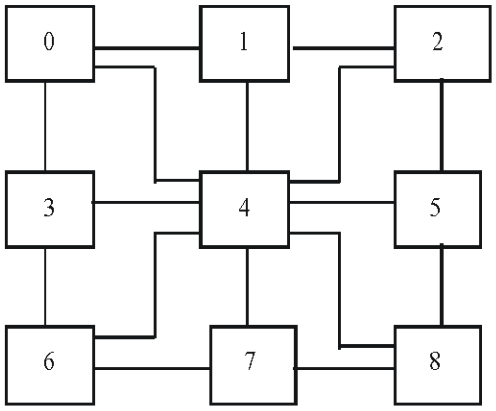

Fig. 1: Interconnection in circuit with 9 cells

The matrix of connections which gathers connections between these cells will thus be symmetrical; the intensity of connections is a function of various parameters such as a number of lines connecting two cells. A significant value means that it has a strong connection between these two cells; a zero value means that there is no connection between them.

The connections matrix of the circuit of the Fig. 1 has for example the following form (Table 1).

Problem states: We can summarize the problem statement of placement in the following way[2]:

Given

K = {0,…,C-1}, a set of cells C_k, k_∈K.

I = {0,…,N-1}, a set of positions ni, i ∈ I.

The connections matrix CM = (cm_{kl}) k,l ∈ K.

We seek a set C of couples (k,i) ∀ k ∈ K with i ∈ I such as

$$\sum_{k\in K}\sum_{l\in K} d_{kl}.\,cm_{lk}$$ That is minimum ∀ k,l ∈ K

∀ i,j ∈ I : dkl = dij with (k,i) ∈ C et (l,j) ∈ C.

We will use the KOHONEN networks[1] to solve this problem and we will proceed as follows:

- A network of N neurons arranged on grid of dimension 2 where each neuron represents a possible site for a cell.

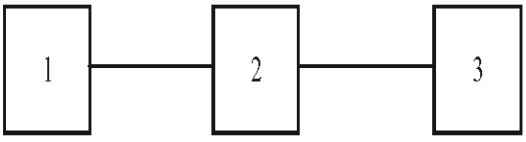

Fig. 2: Implicit Connection

- An input neuron e connected to each neuron of the network.
- A vector stimulus at time t x(t) = ($x_1(t)$, $x_2(t)$, …,$x_e(t)$)
- A synaptic weights vector which characterize a neuron I at time t $w_i(t)$ = ($w_{i1}(t)$, $w_{i2}(t)$,… $w_{ic}(t)$).

Algorithm: The problem is to minimize the energy function, by minimizing the contribution of each c_k cell.

The energy function = $\sum_{k\in K}\sum_{l\in K} d_{kl}\,cm_{lk}$

Contribution of the cell c_k = $\sum_{l\in K} d_{kl}\,cm_{lk}$

The general idea of the algorithm is to place each cell at the best possible position to optimize this sum. Initially, there is a network of empties neurons, after a pretreatment and initialization, a cell is presented to the network to be placed.

An iterative loop is started, with each iteration a cell is presented until we reached a stable state, i.e. until each cell is associated to a neuron which produces the smallest contribution.

Thus, the algorithm of HEMANI and POSTULA which is obtained by refining the corresponding steps of the general KOHONEN algorithm, proceeds in the following 4 steps:

Pretreatment: The goal of the pretreatment is to transform the connections matrix so that, indirect connections which exist between cells which are not connected directly, appear.

Let us suppose that we have a cell c_1 which is strongly connected with two other cells c_2 and c_3 which are not connected between them (Fig. 2); then by transitivity, if the cells are so near to c_1 they are inevitably near one to the other. Thus, it exists an implicit connection between the cell c_2 and c_3.

Hypothesis: Two cells connected to the same cell c_K but not connected between them would be placed at a distance inversely proportional to their intensity of connection to c_K. The distance which allow us to find the strongly connected cell to c_K is given by :

$$dist\,(k,l) = \max(cm_k)/cm_{kl}$$

Algorithm:

Begin

For k=0; k < nmb_cel; k++

For l=k+l; l < nmb_cel; l++

max_d = 0

If $cm_{kl} = 0$ **Then**
For m =0; m < nmb_cel; m++
If $cm_{kl} \neq 0$ and $cm_{ml} \neq 0$ **Then** dist(k,l) = $\max(cm_k)/cm_{km} + \max(cm_m)/cm_{ml}$
If max_d < dist(k,l) **Then** max_d = dist (k,l)
End_If
End_If
End_For
$cm_{kl} = \max(cm_k)/max\_d$
End_If
End_For
End_For
End.

Creation of input sequence: The creation of the input sequence consists to place all the cells the ones after the others while presenting after each cell, the cells which are there connected in the descending order of the connection intensity. The input sequence algorithm of a cell is as following:

Algorithm:
Begin
k = {1,…,c}
k = Random (nmb_cel)
For each cell K **Do**
$Sk = < k,l_1,l_2,l_3,\ldots,l_L >$ such as $cm_{kl1} \geq cm_{kl2} \geq \ldots cm_{klL} > 0$
Sequence = Sequence ⊂ S_k
End.

Where the symbol ⊂ represent the concatenation of two sequences.

Initialization of the synaptic weights matrix: The synaptic weights matrix is initialized by positive low values that we will call Rnd_val.

Iteration: It is the most significant step in this algorithm; we have three main phases[2].

Presentation of a stimulus: We will define time as being a complete treatment of the input sequence. Presentation of a stimulus is done as follows:

⟵ k seq[l]; at beginning time
⟵ k seq[t]; at a time "t"

Response calculation and search for a wining neuron: The minimization of the energy function is obtained by combination effect calculation of the responses and the adaptation law of the synaptic weights.

A good placement for a cell c_K minimize the contribution of this cell, i.e. contribution of

$$c_k = \sum_{l \in K} d_{kl}\, cm_{lk}$$

The response of a neuron must have the following form: Response of neuron n_i with the stimulus

$$c_k = \sum_{l \in K} w_{il}\, cm_{lk}$$

With w_{il} is inversely proportional to d_{il}

The response of each neuron is then obtained by a scalar product between the vector of synaptic weights of the neuron and the vector stimulus presented at the input.

A neuron n_i can give the same response for two different cells c_K and $c_{k'}$, thus it is necessary to determine which of these two cells will occupy this neuron. So, we will put these two cells in competition. The wining cell is given in comparing the response of neuron n_i with the stimuli c_K and $c_{k'}$, making abstraction of the interconnections between these two cells.

The cell which gives the strongest answer is placed out of n_i, The losing cell is removed from the network and synaptic weights matrix is re-initialized with Rnd_val.

Algorithm
Response calculation of each neuron:

For i = 1 to N **Do**
$rik = \sum_{l \in K} w_{ik}\, cm_{lk}$
End_For

File neurons scheduling

The file neuron is defined by a descending order of the answer of each neuron i with the cell k.

In the case of several neurons having the same response for the cell k, we take initially the neuron occupied by the cell k, then the free neurons and finally neurons occupied by the other cells, to prevent that the cell changes unnecessarily the position which it occupies or to pursue the cells whereas there is a free position having an equal answer.

Adaptation of the synaptic weights: After the adaptation, the neurons close to the wining neuron will have a value of weight superior to that of the neurons distant from this one.

After each determination of the wining neuron, the weights vector is updated, so to increase the response of the neuron gaining to the stimulus as well as the neighbors answer.

To illustrate the synaptic weights matrix, a problem of c=4 cells is chosen and a network of 16 neurons, we would like to place these cells in an optimal manner. The weights matrix will take the following form: each neuron is associated to a vector weight (Fig. 4).

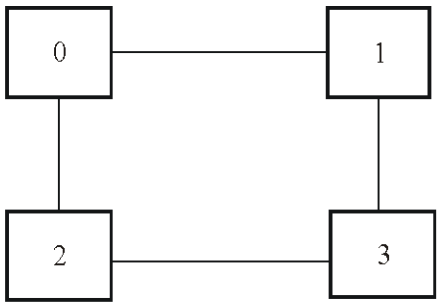

Fig. 3: Connection of 4 cells

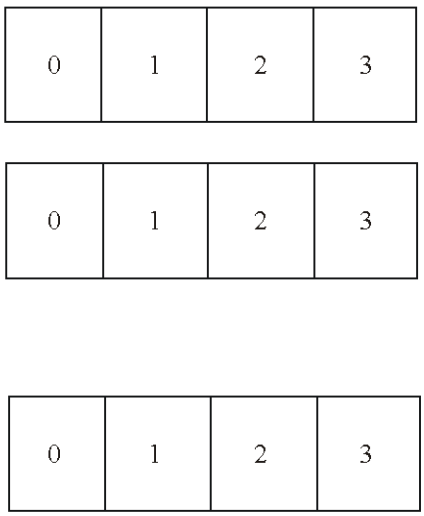

Fig. 4: The synaptic weights Matrix

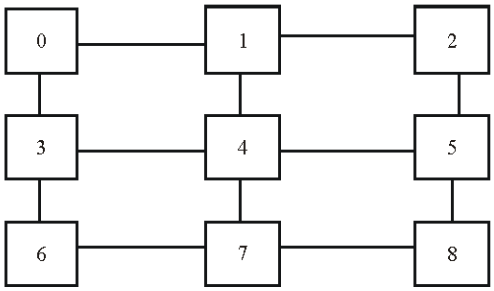

Fig. 5: A connection in circuit with 9 cells

The adaptation weights are done as follows.

Algorithm:
Begin
i: a set of the neurons
For i=1 to N **Do**
If $i \neq \gamma$ **Then**
$w_{ik} = w_{ik} + cost\ \Gamma i\gamma\ (max\ (cm_k)/d_{i\gamma} - w_{ik})$; /* I is not a gaining neuron */
If $i = \gamma$ **Then** /* I is a gaining neuron */
$w_{ik} = w_{ik} + cost\ \Gamma i\gamma\ nb\_p/nmb\_cel\ (max(cm_k) - wi_k)$;
End.

Nb_p : A number of cells placed.
Cost : Static gain (parameter of low value about 0.001).
Γ : Topological function of vicinity.

$$\Gamma_{ij} : \begin{cases} 1, \text{ if dist } (n_i,n_j) \leq vois\_min \\ 0, \text{ if dist } (n_i,n_j) > vois\_min \end{cases}$$

vois_min: minimum vicinity of a neuron n_i which is associated to the cell c_K as being the smallest neighbor which can contain all the cells connected to the cell k; in general it is equal to 2.

Stoping conditions: The iterative loop stops when the placement stabilizes, after trying to place each cell at least once time.

This situation is reached before the convergence of the weights matrix is total. The solution of placement is found, when a fixed position is allotted to each cell.

Example: Let us consider an example of 9 cells illustrated by the Fig. 5 and let us study the algorithm behavior for his case. The interconnections between the cells are given by the following connections matrix (Table2).

The optimal solution is known and corresponds to the diagram of the Fig. 5. The connections matrix, after the pretreatment, becomes (Table 3).

With the input sequence, the cells will take the following form:

Table 2: Connections matrix

0	1	0	1	0	0	0	0	0
1	0	1	0	1	0	0	0	0
0	1	0	0	0	1	0	0	0
1	0	0	0	1	0	1	0	0
0	1	0	1	0	1	0	1	0
0	0	1	0	1	0	0	0	1
0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	1	0	1
0	0	0	0	0	1	0	1	0

Table 3: Connections matrix after the pretreatment

0.0	1.0	0.5	1.0	0.5	0.0	0.5	0.0	0.0
1.0	0.0	1.0	0.5	1.0	0.5	0.0	0.5	0.0
0.5	1.0	0.0	0.0	0.5	1.0	0.0	0.0	0.5
1.0	0.5	0.0	0.0	1.0	0.5	1.0	0.5	0.0
0.5	1.0	0.5	1.0	0.0	1.0	0.5	1.0	0.5
0.0	0.5	1.0	0.5	1.0	0.0	0.0	0.5	1.0
0.5	0.0	0.0	1.0	0.5	0.0	0.0	1.0	0.5
0.0	0.5	0.0	0.5	1.0	0.5	1.0	0.0	1.0
0.0	0.0	0.5	0.0	0.5	1.0	0.5	1.0	0.0

N°	Placement	N°	Placement	N°	Placement	N°	Placement
1	4 -	10	4 1 2 - - - 3 0 5 - - - 7 -	19	4 5 2 - - - 3 4 5 - - - 6 -	28	0 1 2 - - - 3 4 5 - - - 6 7 8 -
2	4 1 -	11	4 1 2 - - - 3 0 5 - - - 6 -	20	- 5 2 - - - 3 4 1 - - - 6 7 0 -	29	0 1 2 - - - 3 3 4 - - - 6 6 7 -
3	4 1 - - - - 3 -	12	4 1 2 - - - 3 0 5 - - - 6 -	21	- 5 2 - - - 3 4 1 - - - 6 7 8 -	30	0 1 2 - - - 3 4 5 - - - 6 7 8 -
4	4 1 - - - - 3 5 -	13	4 1 2 - - - 3 8 5 - - - 6 7 -	22	0 5 2 - - - 3 4 1 - - - 6 7 8 -	31	0 1 2 - - - 3 4 5 - - - 6 7 8 -
5	4 1 - - - - 3 7 -	14	4 1 2 - - - 3 8 5 - - - 6 7 - - - - 0 - - - - - - - - - - - - - - - - -	23	0 5 2 - - - 3 4 1 - - - 6 7 8 -	32	0 1 2 - - - 3 4 5 - - - 6 7 8 -
6	4 1 5 - - - 3 7 -	15	4 1 2 - - - 3 8 5 - - - 6 7 - - - - 0 - - - - - - - - - - - - - - - - -	24	0 1 2 - - - 3 4 - - - - 6 7 8 -	33	0 1 2 - - - 3 4 5 - - - 6 7 8 -
7	4 1 5 - - - 3 0 -	16	4 1 2 - - - 3 8 5 - - - 6 7 - - - - 0 - - - - - - - - - - - - - - - - -	25	0 1 2 - - - 3 4 5 - - - 6 7 8 -	34	0 1 2 - - - 3 4 5 - - - 6 7 8 -
8	4 1 5 - - - 3 0 - - - - 7 -	17	4 1 2 - - - 3 8 5 - - - 6 7 - - - - 0 - - - - - - - - - - - - - - - - -	26	0 1 2 - - - 3 4 5 - - - 6 7 8 -	35	0 1 2 - - - 3 4 5 - - - 6 7 8 -
9	4 1 2 - - - 3 0 - - - - 7 -	18	4 1 2 - - - 3 8 5 - - - 6 7 - - - - 0 - - - - - - - - - - - - - - - - -	27	0 1 2 - - - 3 4 5 - - - 6 7 8 -	36	0 1 2 - - - 3 4 5 - - - 6 7 8 -

Fig. 6: Evolution of Iteration

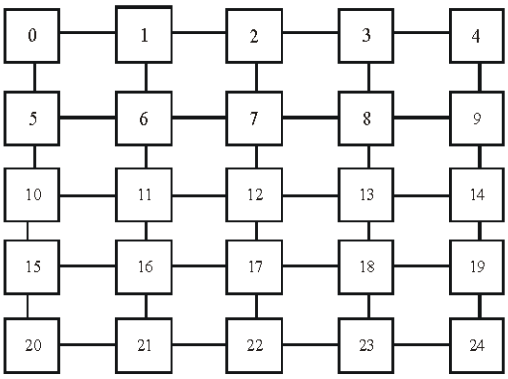

Fig. 7: Optimal solution for a problem of 25 cells

Input Sequence = { 4, 1, 3, 5, 7, 0, 2, 6, 8,
2, 1, 5, 0, 4, 8,
1, 0, 2, 4, 3, 5, 7,
7, 4, 6, 8, 1, 3, 5,
0, 1, 3, 2, 4, 6,
5, 2, 4, 8, 1, 3, 7,
6, 3, 7, 0, 4, 8,
3, 0, 4, 6, 1, 5, 7,
8, 5, 7, 2, 4, 6. }

We will now present the various iterations carried out to reach an optimal solution. The free neurons will be represented by a feature (-), the various cells by their number, the quantity represented in black indicates the stimulus presented in the input (Fig. 6).

Remarks: The first cell presented at the network of neurons is cell 4, we forced it to position with the neuron N°1. Then, the other cells will be presented. The first pursuit occurs with the N°5 iteration, when cell 7 pursues cell 5. Others cells will be presented and others pursuits will occur until iteration N°25, where it is noted that one arrived at an optimal solution. The iterative loop is finished when each one of the cells is presented without a change in the placement.

Tests and results: We will carry out a series of tests on a circuit to evaluate the performances of the algorithm and to draw some conclusions.

We'll evaluate the various results in function of the neural network size, the first cell presented, his position and the various parameters used in the algorithm.
First of all, a circuit of 25 cells will have as an optimal solution the circuit given by the Fig. 7.

Now, we will give the parameters which influence the quality of the placement:

The network size: We carried out a series of tests with two different networks, the first with the same size that the problem, i.e., 5 lines out of 5 columns and the other with 10 lines out of 10 columns. By keeping the same input sequence and the same initial position of the cell at the input, which are determinate randomly, we arrive to:

- An optimal placement on 10 tests for the first network.
- Two optimal placements on 10 tests for the second network.

Remark: It has been noticed that the fact of increasing only the network size does not give a satisfactory results.

Choice of the first cell: In this series of tests, we keep the same networks as before, but now we fix the input sequence, i.e. we run these tests on the same input sequence. The position of the first cell will be given randomly, as done on the first series of tests.

Thus we obtain the following results:

- An optimal placement on 10 tests for the first network.
- Three optimal placements on 10 tests for the second network.

Positions choice of the first cell: We always take the same networks, the input sequence will be obtained randomly and we will position the first cell in the center of the network, Independently of the input sequence. The results of the tests are as follows:

- 0 optimal placement out of 10 tests for the first network.
- 8 optimal placements out of 10 tests for the second network.

ANALYSIS AND ADAPTATIONS OF THE ALGORITHM

In this part, we will try to analyze this algorithm with the study of practical case and to improve it according to the results obtained.

Indeed, according to the results obtained with the algorithm presented, we note for example that the size of the network, the choice of the first cell and its position in the network influence enormously the placement quality.

After several tests carried out, we have encountered problems that we'll detail in the following and we tried to solve these problems, in order to improve the algorithm.

Geometrical limits of neural network: It is noticed that the algorithm proceeds in the following manner: a subset of cells correctly placed ones in relation to the others is amerced, then supplemented until obtaining the complete

-	98	39	19	79	59	69	49	29	89	9
-	95	0	1	2	3	4	5	6	7	8
-	93	10	11	12	13	14	15	16	17	18
-	96	20	21	22	23	24	25	26	27	28
-	94	30	31	32	33	34	35	36	37	38
-	99	40	41	42	43	44	45	46	47	48
-	92	50	51	52	53	54	55	56	57	58
-	97	60	61	62	63	64	65	66	67	68
-	91	70	71	72	73	74	75	76	77	78
-	90	80	81	82	83	84	85	86	87	88

Fig. 8: Geometrical limits of the neuronal network

-	-	-	-	-	-	-	-	-
-	-	50	40	30	20	10	0	-
-	-	-	41	31	21	11	1	-
-	-	-	42	32	22	12	2	-
-	-	-	-	33	23	13	3	-
-	-	-	-	-	24	14	4	-
-	-	-	-	-	-	-	5	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-

Fig. 9: Growth of a solution

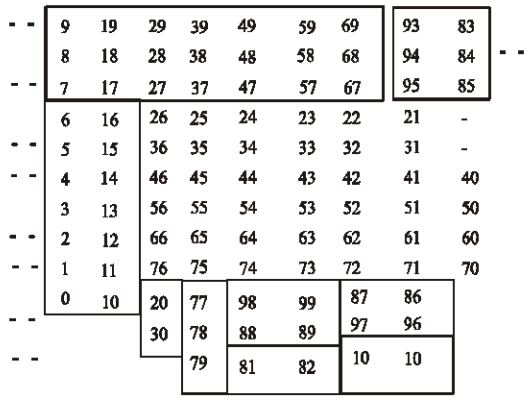

Fig. 10: Growth and collision of several partial solutions

solution. If the solution starts too meadows of one of the network edges, then it is supplemented in direction of this edge, it will not remain enough of place to finish it placement of the others cells. The Fig. 8 shows well the geometrical limits problem.

Growth of the solutions: In spite of that, the problem of geometrical limits can be ovoid by increasing the size of the network, quality of solutions remain weak with cause of another difficulty: When a solution starts in a point of the neural network, the subset of cells correctly placed will grow according to two particular axes to give for example a solution, such as the one represented in the Fig. 9.

Thus when two partial solutions start with two different places, i.e. in the bad place of the network, either these two solutions enter in collision, or they are rather distant one from the other and the total solution is disjoined (Fig. 10). To overcome this problem, we have used the simulated annealing principle[4, 5].

Imrovment of the algorithm using the simulated annealing general concept: When a system is with balance, the expression of the probability that this system may be in a particular state comprises a factor of the form, where E is the total system energy, K the Boltzmann constant and T the temperature[4].

When the temperature of a system constituted of a set of particles is lowered, this system tends to find itself in a state of energy minimal.

We will use the concept of simulated annealing when designing VLSI circuit[5]. The Problems of optimization can often be formulated in an analog form with those searched for a minimal energy state.

Algorithm: We define a variable T representing an artificial temperature. With each iteration, the adaptation of the synaptic weights matrix is done by linear combination between the traditional adaptation and a random term of which the amplitude is decreasing according to the temperature.

Begin

For i=1 to N **Do**

t=t+1

T=1/1+t

If i is a wining neuron **Then**

Adapt = Γiγ (nb_p/nmb_cel) (max(cm_k) - wi_k(t-1));

Else

Adapt = Γiγ (max(cm_k)/ diγ - wi_k(t-1);

noise = rnd_val * T

$w_{ik}(t) = w_{ik}(t-1) + cost[(1-\alpha)\ adapt + \alpha.\ noise]$

End

Where nb_p represents the number of cells placed, T is inversely proportional to the iteration count, α is a parameter of the algorithm.

Global pretreatment: The use of a global pretreatment, i.e., creation of a link between all the cells of the circuit is a very significant solution in the improvement of this algorithm. Thus, a cell is attracted by all the other cells that allow avoiding to have disjoined solutions.

With the first algorithm, cell 28 preserves its position, under influences of cells 29, 33, 34 and 35; whereas cells

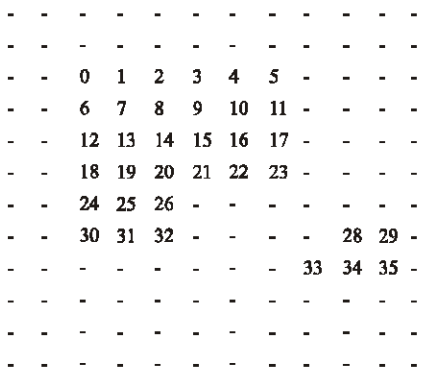

Fig. 11: Attraction of a group on another

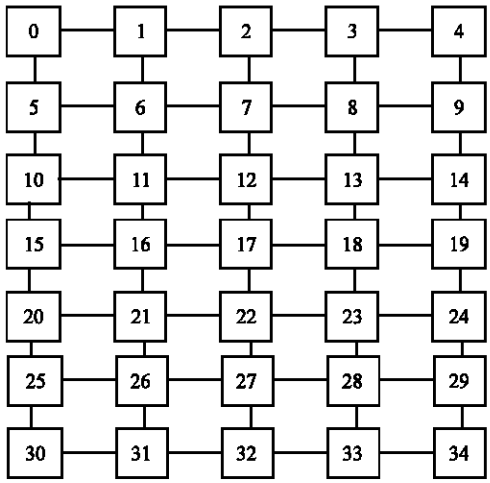

Fig. 12: Optimal solution to a problem of 36 cells

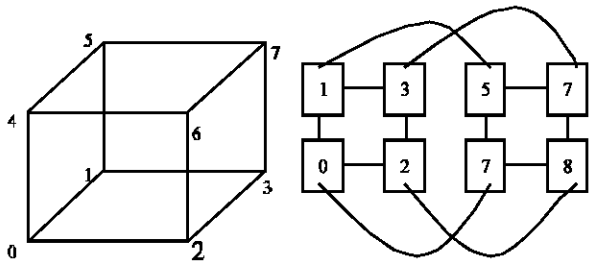

Fig. 13: Hypercube of three dimensions

21, 22, 23, 27 and 32 try to move it. These two forces have roughly the same intensity, therefore cell 28 will not change place.

With the introduction of the global pretreatment, all the other cells which will influence the cell 28 and since their forces are clearly larger than those of cells 29, 33, 34 and 35; cell 28 will change place. This improvement gives good results.

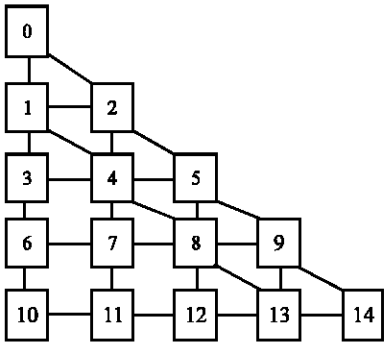

Fig. 14: Triangular problem

Others ameliorations

Network size: To avoid the geometrical limits problem, we take a network sufficiently large, this implies, in general four times larger than size of the problem.

The first cell presented at the input and its position: The first cell presented with the network to be placed, will be selected as being the cell which has a connection with more numbers cells possible and it will be placed in the neuron network centers.

Initialization of the synaptic weights matrix: The synaptic weights matrix isn't initialized by the rnd_val value.

Tests and results: We will test our improved algorithm and adapt it on several problems to evaluate its performances. We will carry out these tests on problems of 25 cells (Fig. 7), 36 cells (Fig. 12) and finally known problem of 100 cells analogue with that of 36 cells. Then, we will carry out a series of tests on a problem which has an hypercube form (Fig. 13), then on an asymmetrical problem (Fig. 14).

Size of the network will be taken four times larger than the size of the problem.

The results of these tests are given in Table 4.

- For the problem of 25 cells, the first input cell is cell 12, it has been placed in the neuron network centers and the optimal solution is obtained after 54 iterations.
- For the problem of 36 cells, the first input cell is the cell 14, it was been placed in the neuron network

Table 4: Results of the test

A number of cells	First input cell	Placement	Iteration count
25	12	Neuron centers	54
36	14	Neuron centers	115
100	44	Neuron centers	211
Hypercube (8)	2	Neuron centers	34
Asymmetric (15)	5	Neuron centers	113

centers and the optimal solution is obtained after 115 iterations.

- For the problem of 100 cells, the first input cell is cell 44, it has been placed in the neuron network centers and the optimal solution is obtained after 211 iterations.
- For the problem of hypercube, we used the circuit of 8 cells, the first cell at the input is cell 2, it was placed in the neuron network centers and optimal solution is obtained after 34 iterations.
- For the asymmetrical problem, we used the circuit of 15 cells, the first cell at the input is cell 5, it was placed in the neuron centers and optimal solution is obtained, by using reduced number implicit connections for the pretreatment, after 113 iterations.

This algorithm, thus made, cannot be applied to problems with greater size, i.e. higher than 100 cells. Thus to solve this problem and understand the solution brought to this difficulty, we should explain the data structure used.

Data stucture: A cell is a record of two fields, the first field indicates a number of real cell connections and the second is a pointer to real which constitute its vector of connection.

```
# typedef struct cell
    int nmb_cel;
    float *ligne_con;
```

A neuron is made up of four fields, its number, its state represented by number of cell that occupy it, its response for the stimulus in entry and finally the last field a pointer to another neuron.

```
# typedef struct neuron
    int numbre;
    int sate;
    float response;
    neuron *neur_next;
```

The matrix of connections is a table of cells: cell mc_c[N], the network of neuron is a chained list structure neuron: neuron * network, the synaptic weights matrix is a pointer to real: float* n_weight.
the size of the problem (the number of cells to place) and the network size (numbers of neurons) used for the placement.

Let nmb_cel be the total number of the cells, li_net et cl_net the number of lines and columns of the network.

The size of these three structures is proportional to We need to allocate (nmb_cel * nmb_cel) real for the matrix of connections, (nmb_cel * li_net * cl_ne) real for the synaptic weights matrix and (li_net*cl_net) neurons to the network.

Table 5: Computing time for various circuits

A number of cells	size network	Iteration count	Computing time
100	20×20	211	1' 15"
144	24×24	302	4' 9"
256	32×32	532	58' 30"
289	34×34	600	1h 6' 55"

Being given that the matrix of connections is symmetrical and of null diagonal. That enabled us to optimize its request for allowance. For each cell i, instead of allocating a table of nmb_cel real, we will only allocates (nmb_cel-i-1) real, which indicates its value of connection with the following cells (i+1, i+2,.., nmb_cel), its value of connection with the preceding cells (0,1… i-1) is in their lines of connections.

At the end of the fundamental case study, a first conclusion may be drawn, the neuronal approach is applicable for VLSI cells placement. The property of classification of the KOHONEN networks is indeed adapted with this type of problems.

Results obtained are now acceptable, though the computing time is significant for problems with big sizes (Table 5)

EXTRAPOLATION OF THE ALGORITHM

Connections between cells: In a circuit, the cells are placed one besides the other to form uninterrupted bands. These bands are placed one above the other and they are separated by one channel (rails of connections) (Fig. 15). Only superiors and inferiors sides of cell are accessible to connections.

Cells of different size: One cell is a functional entity; it varies from one simple logic gate to one unit of multiplication. In one circuit we can find the large and small cells. Small and average cells are gathered in library. In this case the cells are placed by bands; and the height of all cells is identical. Only width cells vary. Very large

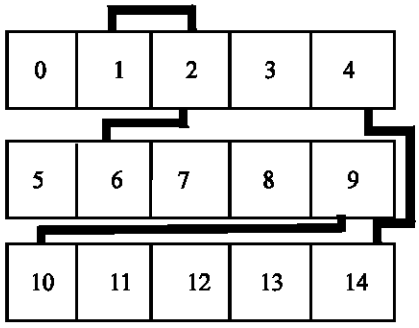

Fig. 15: Cells by bands

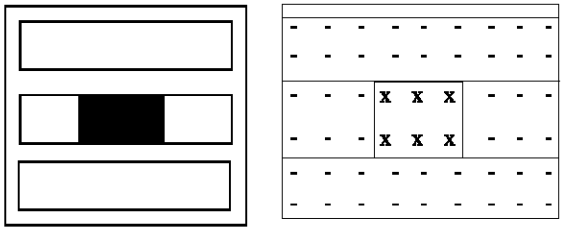

Fig. 16: The cell modeling in a bandage

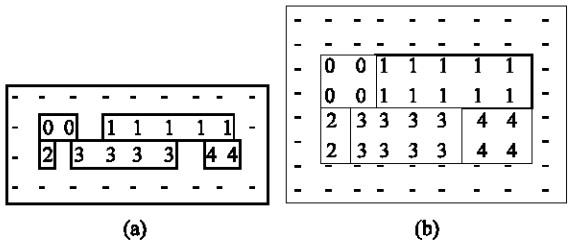

Fig. 17: Neurons bands: (a) with one neuron of high and (b) with two neurons of high

cells are drawn independently of all libraries. They are optimized separately and generally are indicated under term of macro-cells.

Objective seeks: The objective of the algorithm is to minimize length of connection, so to reduce the surface circuit. A surface band is a constant equal to width bands, independently of placement and width zones of connections between bands.

In a circuit connections are realized in using two layers metal (one with the horizontal line, the other, with vertical lines). Height of zone that separate two bands is determined only by maximum numbers of horizontal lines which pass in the same place.

Two methods allow to minimize this height:

- Minimizing the width connections to limit its maximum numbers.
- Supporting connections which don't use the horizontal lines and thus who don't contribute to add horizontal lines.

Problem modeling

Neuronal modeling: For cells of different sizes, we allow a different numbers of neurons and each cell is represented on network by a set of neurons.

Neuronal plan will be divided in neurons bands superposition (Fig. 16).

Structure by bands introduced a constraint as with positions that can occupy a cell in circuit. Only sites located in one bandage are accepted:

- We will not authorize mix of library. All cells have the same height from library where they are exited. All the cells are rectangular. They lay out access for connections on their part higher and lower.
- We don't accept the macro-cells.

Cells modeling: In the extrapolation of the algorithm, we'll introduce concept of limit. Each cell is divided in some numbers of terminals of connection. The numbers of terminals is determined by size of cell and that numbers access is determined by its function.

These terminals of connection represent all possible input of the cell. If all cells have inputs/outputs in only one side, it is essential to specify the side by which connections must join cell. It is necessary in this case to be able to distinguish if one connection has an access high or low of cell, thus two terminals will be necessary for characterize height of cell. Fig. 17 shows cells occupying 1 and 2 neurons.

Adaptation of the algorithm: The introduction of terminals has modified the algorithm, on the one hand we'll work with cells for the input sequence for example and on the other hand we will work with terminals like for placement.

Problem statement: We have a set of cells, each cell is constituted of some numbers of terminals which are connected between them and these links are expressed in matrix of connection between terminals. We ask to place on network a set of terminals confirmedly with topology cells and terminals by minimizing the length of connections. The problem can be formulated as following:

Being given:

$K = \{0,\ldots,C_{-1}\}$, a set of cells c_k
$X = \{0,\ldots,B_{-1}\}$, a set of terminals b_x
$I = \{0,\ldots,N_{-1}\}$, a set of positions n_i

A symmetrical matrix of connections between terminals:

$cm\_b = cm\_b_{ab}$ $a,b \in X$

Thus we can deduce

$K = \{0,\ldots,C_{-1}\}$, a set of cells c_k
$x_k = \{$ limit cell k$\}$

Thus: $X = \bigcup_{k \in K} X_k$

We quest:
A set D of couples (x,i) $\forall$ x $\in$X, i $\in$ I,

Such as : $\sum_{k,l\in K} \sum_{x\in X_k} \sum_{y\in X_L} d_{xy} cm\_b_{xy}$ that is to say minimum

With $\forall$ x,y $\in$X, i,j $\in$ I : dxy = dij with (x,i) $\in$ D et (y,j) $\in$ D.

Pretreatement: The pretreatment is identical with that used in the first algorithm, but with a matrix of connection between terminals rather than a matrix of connection between cells.

Input sequence: Input Sequence is constituted like previously. All cells are introduced in sequence and each one of it is followed by the cells which are connected and it by decreasing order of their intensity connections.
The only modification brought here is following:

When we introduced totality cells in sequence immediately, it is preferable to start by those who are more inter-connected and to allot their center position and to finish with those who have Less connections.

We introduce matrix of connection between cells, calculated from matrix of connection between terminals such as:

$$cm\_c[i,j] = \sum_{x\in X_i} \sum_{y\in X_j} cm\_b_{xy}$$

Connection between cell i and j is equal with summon connections terminals of cell i with terminals cell j.

Responses computation: The aim of the algorithm is to minimize the whole energy function by minimizing each cell rate. The rate of each cell is the sum of the rates of each one of its terminals.

$$r_{ix} = \sum_{y\in X} w_{iy}\, cm\_b_{yx}$$

Since the network response with one cell concern at less two neurons, one for part higher, the other for part lower. It doesn't exist only one gaining neuron but a set of gaining neurons who maximize response of a cell.

With each terminal is associated neuron on which this one is placed. The set of terminal-neurons couples is the set D:
D: set of couples (x, i) x$\in$X et i$\in$I/I set of neurons.

This set can be put in bijection with the unit E cell-neurons couples (Set of couples (k, i) k$\in$K and I$\in$I) where each cell is associated with neuron on which is placed limit corner superior left of cell.

The set D express placement on network of terminals and the set E express it placement on network of cells. It is possible to deduce one from the other.

Let us note that we can give sense to neuron response that if it allots one position acceptable with cell. If the network lines are numbered from zero, only the even lines contain the neurons.

In effect, neurons of the other lines locate cell on two bands. The neurons with the extreme right-hand side plan neuronal place partially the cells in outside network. By convention, answer of these neurons is zero.

In the algorithm proposed in[2], calculation of network response with a cell is carried at following:

$$rik = \sum_{x\in X_k} \sum_{y\in X} w_{iy}\, cm\_b_{xy}$$

Thus answer neuron i for cell k is equal with answer of neuron i for the all terminals of cell k.

After several tests carried out with this method of calculation, we doesn't obtain a good placement because the method takes in consideration only one neuron, then the cell is placed on several neurons and it doesn't' use the totality of the news values founded after adaptations for neurons neighbors of the wining neuron.

We propose a second method of calculation based on first method, but it takes in consideration totality neurons that place cell:

$$rik = \sum_{j\in D_k} r_{jk}\ ;\ D_k \text{ a set of neurons that place cell k.}$$

With:

$$r_{jk} = \sum_{x\in X_k} \sum_{y\in X} w_{jy}\, cm\_b_{xy}$$

This method has given better results, but not the results expected. This method consider place occupied by cell, but it doesn't take count position of limit, because answer of one neuron of the set of the neurons d_K is calculated for the set of terminals x_k.

With this method, the neuron can place only one terminal that conducts us to a third method.

$r_{ik} = \sum_{j\in D_k} \sum_{x\in X_k} r_{jx}$ Such as j places us terminal x of the cell k

With:

$$r_{jx} = \sum_{y\in X_k} w_{jy}\, cm\_b_{xy}$$

Searching the wining neuron: When we have one cell on a network, three cases are possible:

- The place is vacant; in this case the cell place is placed where the response is stronger.
- The space is occupied by the cell; it does not have to change the position.

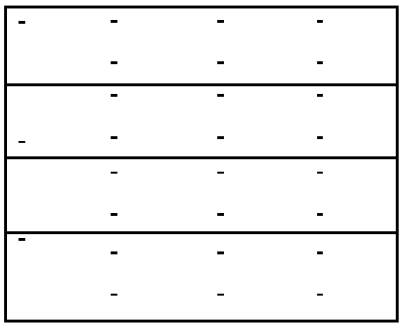

Fig. 18: Neuronal network cut out in bands

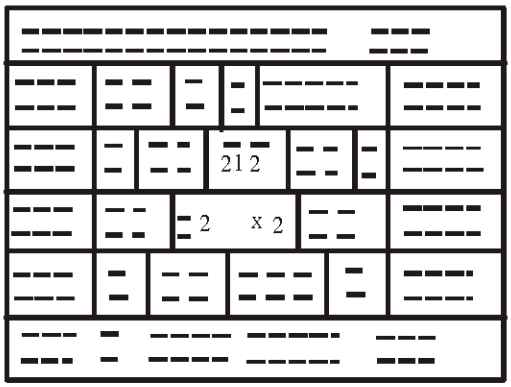

Fig. 19: Circuit placed on network with neighbors neurons of X

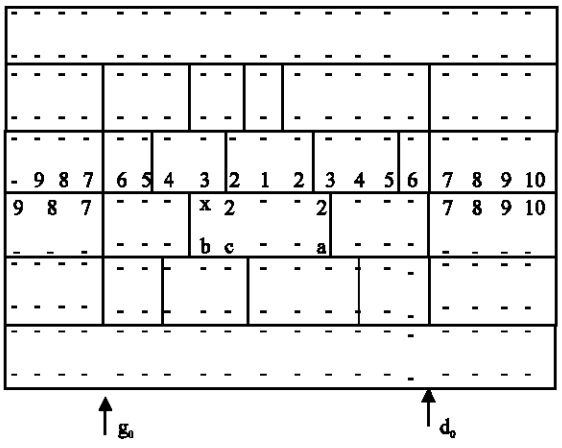

Fig. 20: Distance neurons on both sides of the same connection zone

- The place is occupied by one or several cells; then there is a competition between cells that need this place. The conflict can be solved in three steps:
  - Calculation response of cell that one seeks to place.
  - Calculation of summon answers of the all cells who occupy place coveted.
  - Place will be allotted to the cells having response more strong.

Since the place of a cell not concern only one neuron, we have put a "spaces" function which evaluate degree of occupation of this place and that as follows:

- If the higher left neuron is occupied by cell in input, function space turns zero.
- If the place is occupied by one or several cells, we take the number of terminals higher of each cell, i.e. the unit terminals divided by two and we multiply it by 0.5; then we add 0.25 if one of these cells occupy one part of this place from neuron superior left of the place.

This function allows us to minimize the degree of the circuit deformation at the time of placement, because between two places having the same response and with degree of occupation superior et 0 (i.e. all two are occupied) to put place having more small value of occupation in priority and by consequent, to put in first cell in entry in competition with more small numbers of cell that have small size. Cell driven out of network is immediately placed at the next iteration, that stack LIFO is used.

Adaptation of synaptic weight

New distance: Structure by bands (Fig. 18) of our new algorithm and presence uninterrupted of bands cells can oblige several connections to circumvent some band while passing by rails of connections. This lengthens considerably length of these connections. The evaluation of distance between two neurons must reflect this state of thing.

The distance between two neurons in neuronal plan depends of circuit who occupy network. Let us go to take like example, circuit of Fig. 19, where a terminal x is placed and we want to evaluate distance between all neurons plan and neuron occupied by this terminal.

Figure 19 indicate neurons near to x, It has:

- Only one neuron with distance equal at 1. At this place, the verticals connections don't contribute to increase space between bands. It is better and this translated by a distance more small.

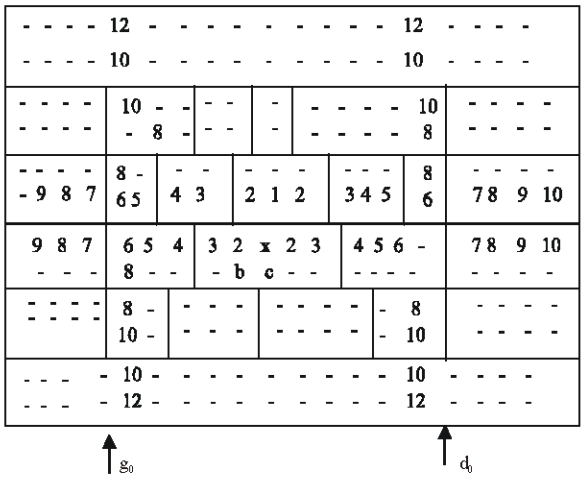

Fig. 21: Distance increase in function of vertical course

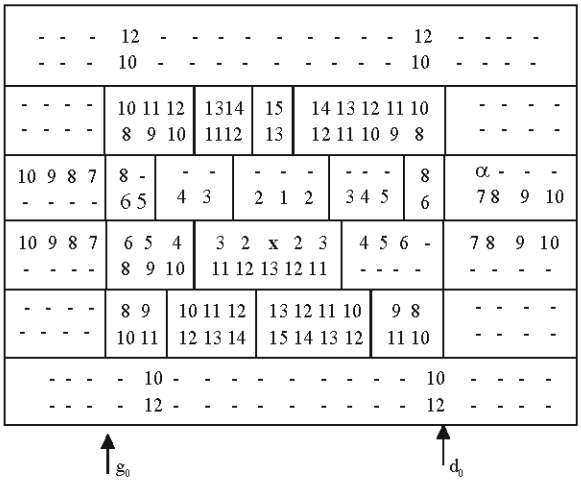

Fig. 22: Distance grows in therefore rails of connection towards interior of circuit

- Four neurons with distance equal at 2, these four positions require one line horizontal to establish connection, position is less good and distance is larger.
- Three neurons a b c, though graphically near, are in fact far. In effect, to connect one limit with x, it is necessary to circumvent band of x.

With this evaluation, the algorithm goes to support connections that don't use horizontal lines and to circumvent bands, thus two manners reduce surface of circuit.

Neurons occupied by terminals positioned on both sides zone of connection who give the access with limit x, are located at distance increasing in deviating of x (Fig. 20).

Arrived in g_0 and d_0, limits of circuit, possible connection can to reach zones of connection in using rail of connection. Distance increase In function of vertical course (Fig. 21).

For neurons occupied by terminals positioned on both sides zone of connections and those who give access with limit x, distance grows regularly while leaving rails of connections towards interior of circuit. When several ways are possible, only more short is taken in consideration (Fig. 22).

The distance between others neurons with neuron x isn't determined. Indeed it is impossible to envisage distance of limit of reference with one of these neurons without know width of cell that will be placed on these . To determine it, one considers that distance increase by one unit, when one deviates circuit while remaining on even line of network. The value of α is initialized of last

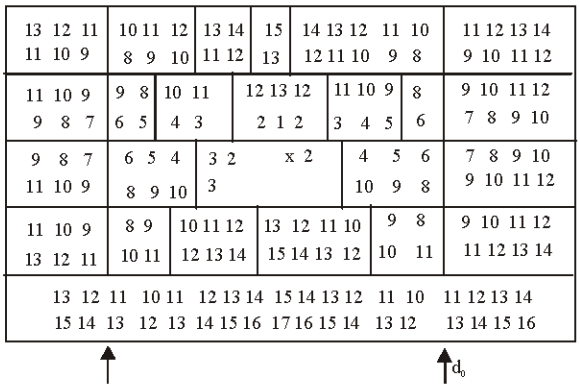

Fig. 23: Distance of others neurons

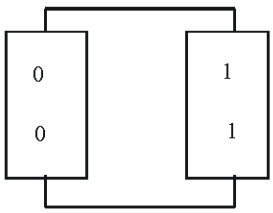

Fig. 24: Circuit of 2 cells and 4 terminals

exact value at interior of the circuit, neurons this give us: α=9. Thus distance of the neurons with neuron x is determined as indicates it in Fig. 23.

The adaptation: The adaptation will hold count of position of each limit of cell who comes to be placed and thus to adapt synaptic weight matrix in taking each time couples (x, y) such as $x \in X_k$ and y place limit x of cell k.

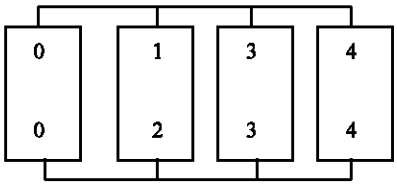

Fig. 25: Circuit of 4 cells and 8 terminals

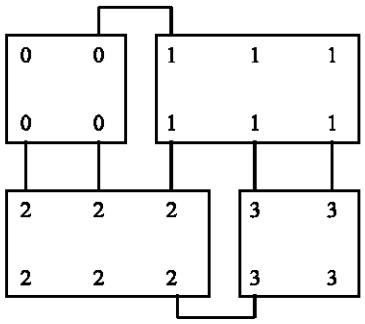

Fig. 26: Circuit of 4 cells and 20 terminals

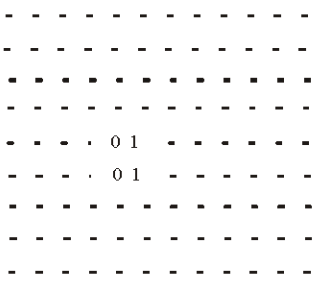

Fig. 27: Solution to the problem of Fig. 24

Algorithm:

Begin

For each limit of cell k **Do**

For i=1 to N **Do**

Begin

t = t + 1 ;

T = 1 / 1+t ;

If i place limit x **Then** /* y = i */

Adapt ← Γ_{iy} (nb_p/ nmb_cel) (max(cm_b_x) - w_{ik}(t - 1) ;

Else

Adapt ← Γ_{iy} (max(cm_b_x) / d_{iy} - w_{ix}(t - 1) ;

noise ← rnd_val * T ;

w_{ix}(t) ← w_{ix}(t-1) + cost[(1-α) adapt + α noise] ;

End

End.

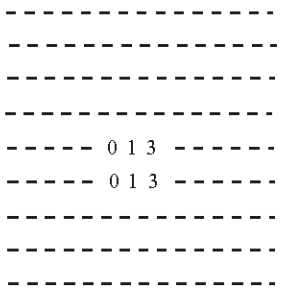

Fig. 28: Solution to the problem of Fig. 25

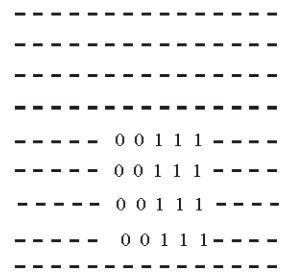

Fig. 29: Solution to the problem of Fig. 26

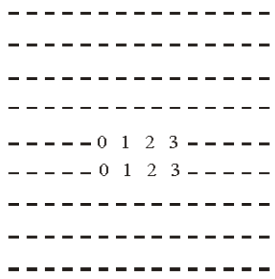

Fig. 30: Solution to the problem of Fig. 25 after deactivation of the pile

Tests and results: We will carry out a series of tests. In first time size of problems to be solved is limited to a small numbers of cells. Fig. 24, 25 and 26 give three circuits that have been used as examples.

From these three circuits, only the first and the third bring program to converge towards a good solution (Fig. 27 and 29). In the second circuit, the algorithm doesn't converge. The program buckles without end. The problem comes that after some iterations, two cells place in even place and one is ejecting the other (Fig. 28).

When cell claims with one place already occupied, the conflict is regulated in determinant with which cell in this space is better appropriate. The cell ejected goes to be replaced owing to stack. It goes certainly to aspire with position that it comes to lose. A new conflict burst but,

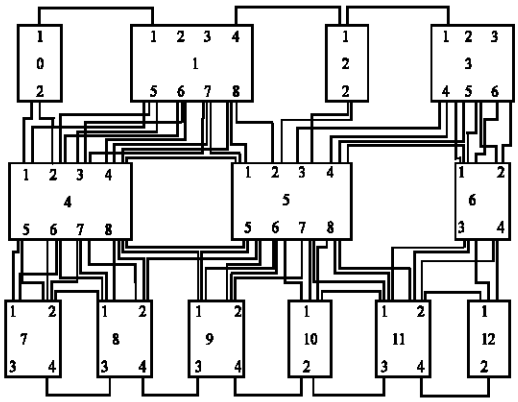

Fig. 31: Circuit of 13 cells and 58 terminals

```
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - 0 1 1 1 1 - - 2 2 2 3 - - -
- - - - 0 1 1 1 1 - - 2 2 2 3 - - -
- - -  4 4 4 4 5 5 5 5 6 6 - - - -
- - -  4 4 4 4 5 5 5 5 6 6 - - - -
- - - 7 7 8 8 - 9 9 10 12 11 11 - - -
- - - 7 7 8 8 - 9 9 10 12 11 11 - - -
- - - - - - - - - - - - - - - - - - -
```

Fig. 32: Solution to the problem of Fig. 31

```
- - - - - - - - - - - - - - - - - - -
 - - - - - - - - - - - - - - - - - -
- - - 0 1 1  1 1 2  2 2 3 - - - - - -
- - - 0 1 1  1 1 2  2 2 3 · - - - - -
- - - 4 4 4  4  5 5  5 5 6 6 - - - -
- - - 4 4 4  4  5 5  5 5 6 6 - - - -
- - - 7 7 8  8  9 9 10 11 11 - 12 - -
- - - 7 7 8  8  9 9 10 11 11 - 12 - -
- - - - - - - - - - - - - - - - - - -
 - - - - - - - - - - - - - - - - - -
```

Fig. 33: Solution to the problem of Fig. 31 without the pretreatment

normally cell ejected must be overcome. In our case it finds its place and the old cell is removed from network.

This cycle continues apparently indefinitely. The solution, more simple, to avoid it problem is to decontaminate stack (Fig. 30).That give us a good placement for circuit of Fig. 28.

Now, we will carry out a test on a circuit where the number of the cells is more significant and connections between the terminals are denser (Fig. 31). The solution of this problem is shown by Fig. 32. It is clear that this solution is bad, by removing the pretreatment we arrives to a better solution (Fig. 33).

DISCUSSION

The search for a gaining neuron and the payment of a conflict remain efficient, even if the placement of a cell is carried out on several neurons. The division of a cell in a unit of terminals is a good idea.

According to the tests which we carried, we noted that the origin of the bad quality of the placement of certain examples is due to the pretreatment. This one is identical to that first algorithm; that it works on the matrix of connection and it takes into account that terminals and not of the cells. It cannot be effective because it doesn't take into account all possibilities of placement and it doesn't have to establish bonds between cells that if those contribute to structure the unit of circuit.

CONCLUSIONS

At the end of the study of fundamental case, a partial conclusion can be shooting. The neuronal approach is applicable for the optimization of the placement of VLSI cells. In spite of restrictive assumptions posed, the starting algorithm gave insufficient results.

We improved this work so that the tests on true circuits can be considered. These improvements related more to obtaining acceptable solutions than to optimize the speed of execution, although the use of the EMS as solution with the problem of insufficiency of the memory capacity, was better than those files in an execution point of view.

With regard to the cells of different sizes, the division of a cell in several terminals seems a good idea. This method must allow widening problems treaties, with the cases circuits comprising macro-cells.

In the last algorithm, we have met problem of no termination because with pretreatment, we cannot envisage its effectiveness for problems of big size.

For future work and to solve the problem of no termination, the pretreatment must take into account all the possibilities of placement and doesn't establish bonds between cells that if those contribute to structure the circuit.

REFERENCE

1. Kohonen, T., 1989. Self-Organization and Associative Memory. 3rd Edn. Springer-Verlag. NY, USA.
2. Hemani, A. and A. Postula, 1990. Postula A cell placement by self-organization neural networks. 3: 377-383.

3. Yildiz, M.C. and P.H. Madden, 2001. Global objectives for standard cell placement ACM Great Lakes Symposium on VLSI, pp: 68-72.
4. Laarhoven, I. *et al.*, 1987. Theory and application. D. Reidel Publishing, Dordrecht, The Netherlands.
5. Guofang, N. *et al.*, 2005. Adaptive Simulated Annealing for Standard Cell Placement. ICNC, 943-947.