A Neural Network Approach to the Placement Problem

M. Saheb Zamani

School of Computer Sc. and Eng.
University of N.S.W.
Sydney, NSW, 2052
Tel: +61 2 385-4898
Fax: +61 2 385-5995
e-mail: morteza@vast.unsw.edu.au

G.R. Hellestrand

School of Computer Sc. and Eng.
University of N.S.W.
Sydney, NSW, 2052
Tel: +61 2 385-4028
Fax: +61 2 385-5514
e-mail: G.Hellestrand@unsw.edu.au

Abstract— In this paper, we introduce a new neural network approach to the placement of gate array designs. The network used is a Kohonen self-organising map. An abstract specification of the design is converted to a set of appropriate input vectors fed to the network at random. At the end of the process, the map shows a 2-dimensional plane of the design in which the modules with higher connectivity are placed adjacent to each other, hence minimising total connection length in the design. The approach can consider external connections and is able to place modules in a rectilinear boundary. These features makes the approach capable of being used in hierarchical floorplanning algorithms.

I. Introduction

Given an abstract specification of a circuit, a placement algorithm finds the position of all modules in the design in a 2-dimensional plane so that an objective function (usually total connection length) is minimised. The specification normally contains a list of connectivities[1] between the modules in the form of a matrix, called a connectivity matrix and sometimes restrictions on the module port positions.

In this paper, we present an algorithm based on the Kohonen self-organising map [4]. The self-organising principle has been previously applied to the placement problem [3, 2, 6, 1]. Kim and Kyung [3] take the two coordinates, (x, y), of the modules on a plane as the inputs to the network and the modules themselves as the outputs. In this way, each neuron (module) is associated with the two coordinates (x, y) at which it is placed. The neighbourhood of an output neuron (module) is defined as the neurons corresponding to the modules having connections to it. The process involves applying a random (x, y) coordinate pair to the network at each iteration, selecting the module which is the least distance from the coordinates in the input vector, and updating the coordinates of the neighbouring nodes according to their current distance from the input vector. After a number of iterations, the algorithm converges to the state where the positions of all modules are determined. Chang and Hsiao's approach [1] is similar to the above except they use a force-directed method to generate the input vectors based on the current position of the modules. These approaches do not distinguish between high and low connectivities between modules, hence the neighbourhood may inappropriately include some modules with low connectivity.

Hemani and Postula's approach [2] is different in that the inputs are n dimensional vectors originating from the connectivity matrix (the coordinate j of an input vector X_i, $x_{i,j}$, is the number of connections between module i and module j). The neurons form a 2-dimensional plane where the modules are to be placed. During the process, the vectors are input to the network and the weights are updated so that the neighbouring neurons of the module associated with the input vector become more attractive to the modules connected to it. At the end of this process, each neuron is occupied by a module in such a way that the connected modules are placed close together. This approach works well for a binary connectivity matrix but does not converge in many cases [6].

In our approach, a set of n-dimensional vectors is fed to the network and the map assigns the modules to the slots (neurons) in an optimised way. Connectivities greater than 1 are also considered in the process. The idea is explained in detail in Section III after a brief introduction of self-organising neural networks.

II. Self-Organising Maps

A self-organising neural network consists of m output nodes which are fully connected to n inputs with variable weights (Fig. 1). Therefore, each output node j has an n-dimensional weight vector $\vec{W}_j = (w_{0j}, w_{1j}, ..., w_{n-1,j})$. The weight vectors are set to small random numbers initially and are updated at each iteration. Each time, an n-dimensional input vector $\vec{V} = (v_0, v_1, ..., v_{n-1})$ is applied

[1] Connectivity is often considered as the number of connections between modules. It can also be defined as a number indicating the importance of connections to be minimised.

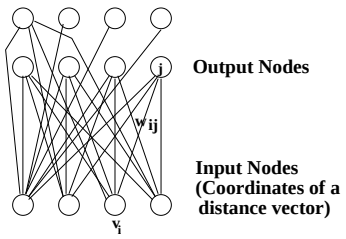

Fig. 1. Self-organising neural network.

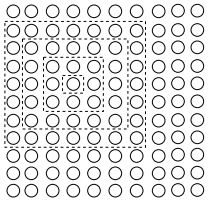

Fig. 2. Neighbourhood in the self-organising map.

to the network and all output nodes compare their weight vectors with the input vector. The node whose weight vector is most similar to the input vector (*i.e.* their Euclidean distance of the two vectors is the least) is selected and its weight vector together with the weight vectors of all its neighbouring nodes are updated so that they become more similar to the input vector by a gain factor (η):

$$w_{ij_{new}} = w_{ij_{old}} + \eta(v_i - w_{ij_{old}}). \qquad (1)$$

The neighbourhood covers most of the output nodes around the selected node at the beginning but it is reduced during the process until it contains only the selected node (Fig 2). The gain factor is set to a number less than and close to 1 initially and decreases linearly in time. At the end of the process (training the network), the output nodes which are geographically close in the network are assigned by (*i.e.* respond best to) the input vectors with small distances and the input vectors with large distances are assigned to far output nodes.

III. Basic Concept

A set of n nodes are to be assigned to n modules which are connected by a number of wires. The connectivity information is given in the form of a matrix whose entry, C_{ij} $i, j \in [0, n-1]$, is the connectivity between the module pair i and j. To minimise the total connection length in the design, heavily connected modules should be placed adjacent to each other. Since in general, satisfying all the adjacency requirements may not be possible in a 2-dimensional plane, a self-organising map is used to map this higher dimensional proximity information onto a 2-dimensional plane.

The Kohonen self-organising process starts with a set of input vectors representing module connectivities and assigns the vectors with highest connectivities (least distances) to the neurons which are geographically close in the map after a number of iterations. To apply the self-organising principle to the placement problem, we calculate a vector for each module so that the distance between each pair of modules reflects the final desired proximity of the module pair in the 2-dimensional map. The desired proximity can be derived from the connectivity information; the heavier the connectivity C_{ij}, the closer (in distance) the modules i and j should be placed. These vectors are then fed to the network and recycled at random during successive iterations. At the end of the process, the distance vectors belonging to the modules heavily connected are assigned to the neurons which are geographically close.

For hierarchically specified circuits, the floorplan of the upper hierarchy levels can guide the floorplanning process of the lower levels by applying constraints to the submodules' ports positions [8]. These constraints are considered in our algorithm by initialising the weight vectors of the neurons at the desired port positions to values close to the main vectors of the submodules connected to the ports, depending on the number of connections between them. In other words, the ports are considered as single point submodules whose positions are fixed. In this way, the algorithm can handle complex cases where a port is connected to many modules with different connectivities and can place the modules with more connections closer to that port.

Another strength of this approach is its capability to work with rectilinear shapes. The complexity of rectilinear shapes results in current floorplanners employing rectangular dissection approaches which may result in large waste space. In our approach, this is done simply by considering a rectilinear map with the shape obtained from the upper level floorplanning process.

IV. Distance Vectors

The objective is to generate a set of vectors $\vec{V}$ whose similarities (distances) correspond to the connectivities between n modules. The following equation holds for all vector pairs $\vec{V_i} = (v_{0i}, v_{1i}, ..., v_{n-1,i})$ and $\vec{V_j} = (v_{0j}, v_{1j}, ..., v_{n-1,j})$:

$$\sum_{k=0}^{n-1}(v_{ik} - v_{jk})^2 > \sum_{k=0}^{n-1}(v_{ik} - v_{lk})^2 \qquad \textit{iff} \quad C_{ij} < C_{il}.$$

If the vectors are normalised vectors, one possible solution to the above relations is:

$$\forall i, j = 0, ..., n-1$$

$$\vec{V_i}.\vec{V_j} = K.C_{ij} < 1 \quad if \quad i \neq j$$
$$\vec{V_i}.\vec{V_i} = 1 \quad if \quad i = j \quad (2)$$

where K is a constant factor to keep the v_{ij}'s within the domain of real numbers[2]. This is because when

$$\sum_{k=0}^{n-1}(v_{ik}.v_{jk}) < \sum_{k=0}^{n-1}(v_{ik}.v_{lk})$$

then

$$\sum_{k=0}^{n-1}(v_{ik}-v_{jk})^2 > \sum_{k=0}^{n-1}(v_{ik}-v_{lk})^2$$

for normalised vectors due to the fact that

$$\sum_{k=0}^{n-1}v_{ik}^2 = \sum_{k=0}^{n-1}v_{jk}^2 = \sum_{k=0}^{n-1}v_{lk}^2 = 1.$$

This results in a system of $\frac{n(n+1)}{2}$ equations with n^2 variables, so we can select the $\frac{n(n-1)}{2}$ variables arbitrarily to generate a solution. To simplify the solution process, we take

$$v_{ij} = 0 \qquad \forall i > j.$$

V. Algorithm

After initialising the weight vectors for all neurons to small values, the vectors generated by the method of Section IV are selected at random and fed to the network at each iteration. As in the standard Kohonen self-organising process [4], the best responding neuron j^* corresponding to the minimum d_j is selected where:

$$d_j = \sum_{i=0}^{n-1}(v_i - w_{ij})^2,$$

v_i = the ith coordinate of the distance vector,

w_{ij} = the weight from the input node i to the output node j (see Fig. 1).

Then the weights of the neighbouring nodes to j^* are updated according to equation 1.

The above procedure is repeated for a specified number of iterations. As with the standard self-organising process, convergence is guaranteed [4].

VI. Experimental Results

The algorithm was applied to 4 examples derived from [5] and [2]. Only the global ordering phase [4] was used to train the network. The initial neighbourhood was set so as to cover about 70% of the neurons, and then the coverage was decreased linearly with time to 0 after 1000 iterations. The gain factor was initialised to 0.4 and decreased linearly with time to 0.04 after 1000 iterations, as recommended by Kohonen [4]. The network size was set to 4 times the number of modules to avoid assignment of more than one module to one node.

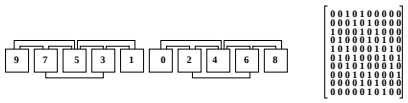

Fig. 3. 10-module problem.(a): An optimal placement (b): Connectivity matrix.

For the 10-module example described in [5] (Fig. 3), the Persky and Smith's algorithm [5] which is based on a Hopfield neural network produced more than 30% worse results than the optimum solution. The optimal solution's total connection length is 20 for this example. This result was achieved in 50 trials in which only 5 of the trials were valid.

We applied our algorithm to this problem using 4 trials all of which were valid and very close to the global optimum (Table I). The algorithm in [2] produced optimal result for this example, but may not work for the examples with non-binary connectivity matrices. We changed the connectivities between the modules 1 and 3, and between modules 3 and 5 to 16 to compare our algorithm with [2]. While our algorithm gave a result not far from optimum, the algorithm in [2] did not converge after 5000 iterations even for a number of trials with different initial weights.

The second example, adopted from [2] is known as the 9-module design (Fig. 4). Although the approach in [2] sometimes produces optimum results, it does not converge in many cases even for the binary connectivity matrix [6]. In 3 trials, our algorithm produced 3 valid results one of which equal to the global optimum and the other two close to optimal solution (Table I). We again changed the connectivity matrix so as to contain non-binary values (Fig. 5). As shown in Fig. 6, our approach has placed all the modules with high connectivities (solid lines) adjacent to each other and other adjacency requirements (dotted lines) are also met to some extent. A row assignment method as in [3] may be applied to modify the module positions. The algorithm in [2] did not converge for this example after 5000 iterations.

The time required by the algorithm on a SUN4 workstation for all the above examples was less than 5 seconds which shows the approach is practical. The summary of the results is given in Table I.

[2] K must be chosen so as to make v_{ik} small enough to satisfy equation 2.

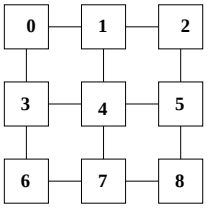

Fig. 4. 9-module example.

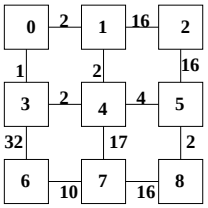

Fig. 5. 9-module example with non-binary connectivities.

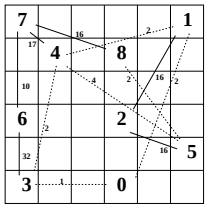

Fig. 6. The result of placement for the modified 9-module example.

TABLE I
COMPARISON OF CONNECTION LENGTH FOR 4 EXAMPLES.

		Total connection length			
	No of modules	Persky's Alg.	Hemani's Alg.	Our Alg.	Opt. Sol'n
Persky example	10	27 (mean)	20	22 (mean)	20
Persky ex. (non-binary)	10	-	Not Converged	70	50
Hemani example	9	-	12	17.7 (mean)	12
Hemani ex. (non-binary)	9	-	Not Converged	see Fig 6	-

VII. Conclusion

In this paper, we presented an approach based on self-organising maps for the placement of regular designs like gate arrays. The proximity requirements, in the form of a connectivity matrix, which may not be fully met in 2 dimensions were converted to a set of n-dimensional distance vectors whose similarities (distances) correspond to proximities. These vectors were then mapped to a 2-dimensional plane by the Kohonen self-organising method. The advantage of this approach over existing approaches is that it considers connectivities greater than one in the calculations, as is required for real designs. The algorithm always gave valid results very close to optimum and converged in short periods of time. The approach is capable of placing the modules inside rectilinear regions with external ports connection constraint, which may be useful in hierarchical floorplanning. Another version of the algorithm has been successfully applied to the floorplanning problem [7].

References

[1] R-I Chang and P-Y Hsiao. Arbitrarily sized cell placement by self-organizing neural networks. *Proceedings of IEEE International Symposium on Circuits and Systems*, pp. 2043–2046, 1993.

[2] A. Hemani and A. Postula. Cell placement by self-organisation. *Neural Networks*, 3:377–383, 1990.

[3] S-S Kim and C-M Kyung. Circuit placement on arbitrarily shaped regions using the self-organization principle. *IEEE Transactions on Computer-Aided Design*, 11(7):844–854, July 1992.

[4] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, September 1990.

[5] G. Persky and W. R. Smith. Experiments on cell placement with a simulated neural network. *Proceedings of International Workshop on Placement and routing*, pp. 7.4–7.7, 1988.

[6] R. Sadananda and A. Shrestha. Topological maps for VLSI placement. *International Joint Conference on Neural Networks*, pp. 1955–1958, 1993.

[7] M. Saheb Zamani and G. R. Hellestrand. A new neural network approach to the floorplanning of hierarchical VLSI designs. *International Conference on Neural, Parallel and Scientific Computations*, 1995.

[8] M. Saheb Zamani and G. R. Hellestrand. A stepwise refinement algorithm for integrated floorplanning, placement and routing of hierarchical designs. *Proceedings of IEEE International Symposium on Circuits and Systems*, 1995.