Optimized Local Relocation for VLSI Circuit Modification Using Mean-Field Annealing

Gholam Reza Karimi, Ahmad Azizi Verki, and Sattar Mirzakuchaki

In this paper, a fast migration method is proposed. Our method executes local relocation on a model placement where an additional module is added to it for modification with a minimum number of displacements. This method is based on mean-field annealing (MFA), which produces a solution as reliable as a previously used method called simulated annealing. The proposed method requires substantially less time and hardware, and it is less sensitive to the initial and final temperatures. In addition, the solution runtime is mostly independent of the size and complexity of the input model placement. Our proposed MFA algorithm is optimized by enabling module rotation inside an energy function called permissible distances preservation energy. This, in turn, allows more options in moving the engaged modules. Finally, a three-phase cooling process governs the convergence of problem variables called neurons or spins.

Keywords: VLSA, MFA, simulated annealing, cooling process, relocation.

I. Introduction

In modern VLSI physical design, engineering change order optimization methods are used to mitigate model placement problems, such as hot spots, that are identified in a given layout during post-routing analysis. Similarity of model placement and the resulting placement are essential, and the speed of the migration method is also important.

In the simulated-annealing (SA)–based method proposed in [1] to resolve the placement migration problem, an additional module is added to a model placement [1]. We focused on modifying this method by inserting an extra module into a model placement based on a neurocomputing method, mean-field annealing (MFA).

Neurocomputing approaches based on the Hopfield model have been successfully applied to various combinatorial optimization problems, such as the traveling-salesman problem [2], [3], the scheduling problem [4], the mapping problem [5], the knapsack problem [6], [7], the communication-routing problem [8], the graph-partitioning problem [3], [9], [10], the graph layout problem [11], and the circuit partitioning problem [12], [13].

In [14], the MFA algorithm was applied to the cell placement problem and the MFA algorithm was demonstrated to be faster than the APR algorithm which uses the SA method as its core algorithm. Relations of our MFA relocation algorithm are similar to those of the MFA algorithm proposed in [14]. In addition, we defined a new energy function, namely, permissible-distances-preservation energy, to keep the boundaries of the local relocation region fixed. The ability to rotate modules is added to MFA algorithm, and an applied cooling process is carried out in three phases.

The rest of this paper is organized as follows. In section II, we

Manuscript received Sept. 9, 2009; revised July 28, 2010; accepted Aug. 17, 2010.

Gholam Reza Karimi (phone: +98 831 4274536, email: ghkarimi@razi.ac.ir) and Ahmad Azizi Verki (email: aziziverki@gmail.com) are with the Electrical Engineering Department, Razi University, Kermanshah, Iran.

Sattar Mirzakuchaki (email: m_kuchaki@iust.ac.ir) is with the College of Electrical Engineering, Iran University of Science and Technology, Tehran, Iran.

doi:10.4218/etrij.10.0109.0533

introduce the stages of our local relocation problem. In section III, we explain the information that has to be extracted from a relocation range and its calculations. Our MFA algorithm structure is defined in section IV. In section V, we present our experimental results, and we conclude this paper in section VI.

II. Relocation Problem and Local Relocation Algorithm

Our relocation problem is formulated as follows:

- Input: Model placement, including a set of modules and a netlist or hypergraph representation of the circuit, the additional module with its coordinates, and the incident nets,
- Output: Local relocated placement,
- Objective: Fast relocation with a minimum number of displacements and greater similarity,
- Constraint: No overlap between modules and preservation of permissible distances.

1. Relocation Problem

There are four classified approaches to the problem of inserting an extra module into a model placement that are referred to in [1]:

i) The additional elements are inserted into unoccupied "whitespace" areas as much as possible.
ii) Before additional logic elements are inserted, an effort is made to predict the amount of whitespace area required; this whitespace is distributed over the chip. If the prediction is accurate (or conservative), the added elements can be placed within the available space.
iii) The third approach is to simply insert or resize the required logic elements and begin the optimization process from scratch.
iv) The fourth approach is to insert additional logic elements without considering overlaps.

We adopt the fourth of these approaches. The MFA relocation algorithm removes overlaps by moving or rotating modules. Note that all of the movements and rotations must observe some permissible distances, as we will explain in the following sections. The feasibility of solving a problem depends on the topology of placement and similarity. It is clear that selecting a large area of a model placement as the relocation range may allow a feasible solution, but it leads to more dissimilarity.

2. Local Relocation Algorithm

The proposed relocation algorithm consists of two stages: first, the construction of MFA vectors and calculation of permissible distances from a proper relocation range around an additional module; second, local relocation with MFA. In the first stage, given the model placement and an additional module with its coordinates, the environment space around the additional module is scanned to find the proper range that has enough free space as the local relocation range. Then, information such as movable and fixed modules and the distances between fixed modules are extracted. In the second stage, the MFA algorithm starts to move or rotate movable modules considering critical distance criteria using outputs of the first stage.

III. Calculation of Permissible Distances and Construction of MFA Vectors

The MFA relocation problem consists of a hypergraph $\Omega(C, N)$, representing the circuit to be relocated and a rectangular grid of clusters with P rows and Q columns as the relocation range. Hypergraph $\Omega(C, N)$ consists of a vertex set C, representing the modules of the circuit, and a hyperedge set N, representing the nets of the circuit. The first stage of the local relocation algorithm has to extract the hypergraph representation information of the selected part of the model placement as inputs of the second stage, such as P, Q, sets C and N, and MFA input vectors.

The part of the model placement selected as the local relocation range must have enough free space or dead space to allow insertion of an extra module.

Selecting the size and position of the relocation range depends on the size of the additional module and the desirable similarity between the model placement and the relocated placement. It is clear that selecting a larger part of a model placement as a relocation range may cause more dissimilarity; therefore, this algorithm searches around the additional module in various directions considering relocation range limitation to find the desirable range.

After the relocation range is determined, its underlying modules are classified into two groups. The first group includes modules that are completely inside the relocation range and are movable. The second group consists of modules that just overlap with the relocation range and must have fixed positions during relocation because they form a frame around the movable modules.

Actually, if we think of the model placement as a puzzle, this frame is just a piece of it. It is clear that, after local relocation, this piece must fit in its location again; therefore, any movement or rotation of inside modules must preserve vertical and horizontal distances between outer modules. Figure 1(a) shows the relocation range and its underlying modules in the model placement. Figure 1(b) shows the local relocated placement of Fig. 1(a).

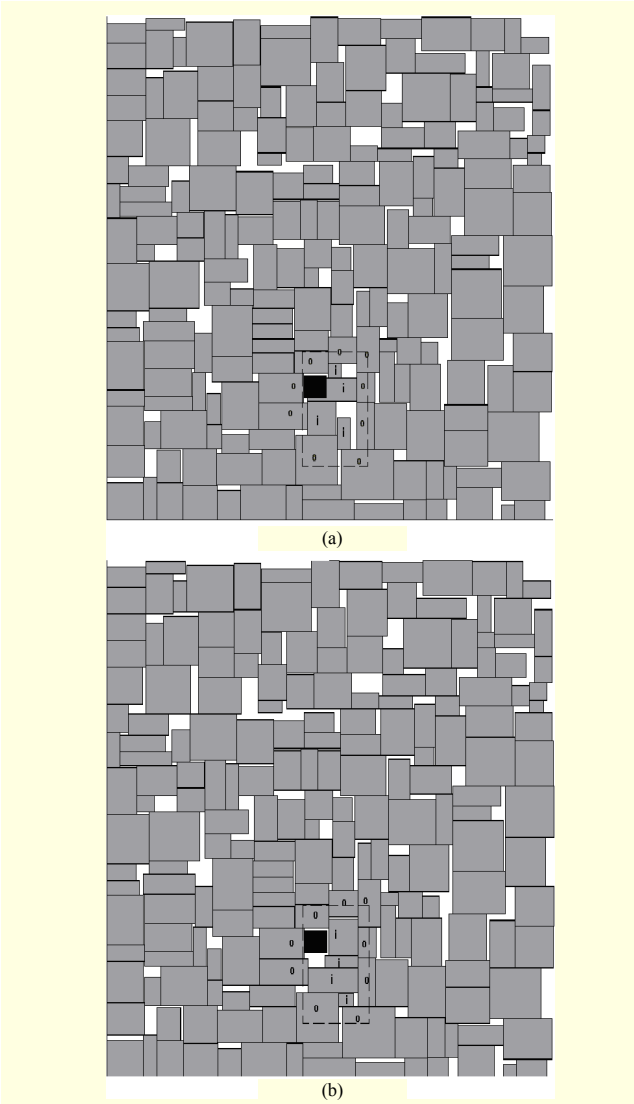

Fig. 1. a) Relocation range and its underlying module and b) local relocated placement using MFA.

The dashed square in Fig. 1 shows the relocation range, and the black module is the additional module. Modules marked as "o" are outer modules, and those marked as "i" are inner modules. In our method, we use MFA with a discrete variable for relocation, so the configuration of the problem must encode to the discrete space. As a result, the width and height of the relocation range are divided into equal spans that form columns and rows, respectively. The rows and columns that are occupied with modules are marked. The outer modules are then separated into four sets: upper, lower, left, and right boundary modules.

1. Calculating Permissible Distances

For each row or column, two modules are determined as its boundary modules. The permissible distance of each row or column is obtained by calculating the distance between the left and right boundary modules of that row or the distance between the upper and lower boundary modules of that column, respectively. Figure 2(a) shows the coordinates of a module. The lower-left and upper-right corners of a module are considered here. The upper-right corner coordinates of module "*i*" are obtained by

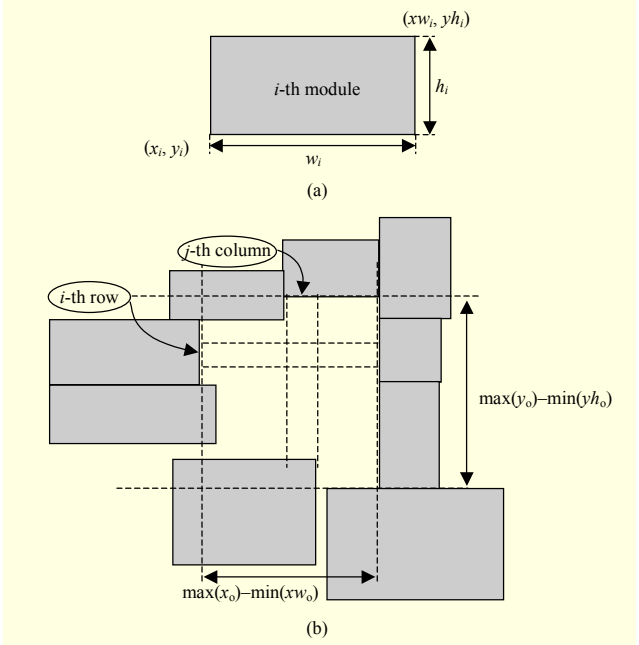

Fig. 2. a) Coordinates of module and b) boundary modules.

$$xw_i = x_i + w_i, \quad yh_i = y_i + h_i. \tag{1}$$

For each row or column, two modules are determined as its boundary modules. Figure 2(b) shows the boundary modules of the relocation range shown in Fig. 1.

In Fig. 2(b), considering the coordinates of the boundary modules of that row or column, row and column permissible distances are computed by

$$Rpd_i = x_{oi} - xw_{oi}, \quad Cpd_j = y_{oj} - yh_{oj}. \tag{2}$$

Subscript "o" denotes outer modules, and Rpd_i and Cpd_j represent the i-th row's and j-th column's permissible distances, respectively.

In the main algorithm, the sums of widths or heights of modules that are located in the same row or column are calculated, and the results are not permitted to exceed the permissible distance of each given row or column. To reduce the numbers of variables and calculations, the outer modules, which must have fixed positions, are excluded, and only the inner modules, which are movable, are entered into the MFA algorithm.

2. Construction of MFA Initial Average Spin Vectors Based on Positions of Movable Modules (Mapping)

As previously mentioned, the extra module, as an overlapping module, is entered into the algorithm, but it remains in its location during the algorithm. Some outer modules that protrude into the inner module area could be entered into the MFA algorithm to prevent some undesirable relocating. We divide the inner module area into P rows and Q columns.

The minimum heights and widths of the modules are obtained. Then, the width and height of the relocation range are divided into these obtained values and rounded to integers which are the numbers of columns and rows, that is, Q and P.

We define the position of a module using two vectors in the MFA space, one representing its vertical position and another representing its horizontal position. These vectors have P and Q elements, respectively. For module m, these vectors are shown with v_m^r and v_m^c, which finally form overall matrices, which are denoted as v^r and v^c. Each element of these vectors is called a *spin* (neuron), and the sum of the values of these elements is 1.

The lower-left corner coordinate of a module determines its position. This means that, if this point is located in the range of the i-th row and the j-th column, the i-th element of v_m^r and the j-th element of v_m^c are set to 1, and the others are set to 0:

$$v_m^r = [\underbrace{0 \quad \ldots}_{1:i-1} \; \underbrace{1}_{i} \; \underbrace{\ldots \quad 0}_{i+1:P}], \quad v_m^c = [\underbrace{0 \quad \ldots}_{1:j-1} \; \underbrace{1}_{j} \; \underbrace{\ldots \quad 0}_{j+1:Q}]. \tag{3}$$

To construct precise vertical and horizontal vectors, we use a pseudo-trigonometric method. Module position is determined using its lower-left corner distance with the lower-left corner of the relocation range with coordinates (x_{rr}, y_{rr}).

Figure 3 shows the relocation range of Fig. 1 and its incident inner modules, which are shown in a darker shade of gray. We use a special value to normalize these distances. This value is the Euclidean distance between the lower-left corner of the relocation range and a point at the coordinates of the inner modules maximum x and maximum y:

$$Sd = \sqrt{(\max(x_{in}) - x_{rr})^2 + (\max(y_{in}) - y_{rr})^2}\,. \tag{4}$$

Then, to calculate the row vector of a module, its vertical distance with the lower-left corner of the relocation range is obtained and then normalized as

$$vd_i = \frac{y_i - y_{rr}}{Sd}, \qquad hd_j = \frac{x_j - x_{rr}}{Sd}. \tag{5}$$

The same calculation is carried out for the column vector.

The normalized total horizontal and vertical ranges are

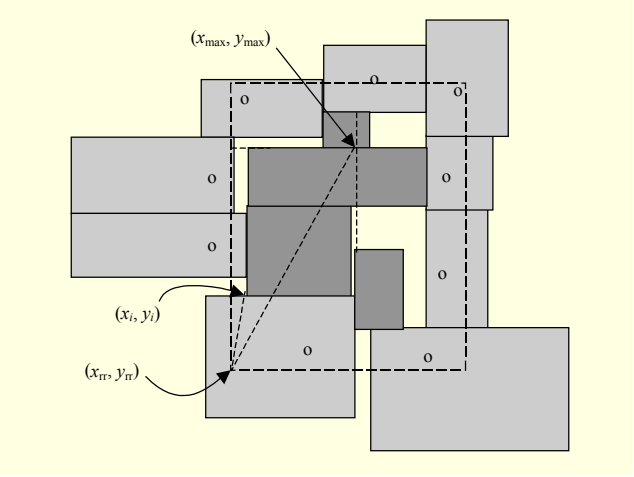

Fig. 3. Relocation range and its inner modules for construction of MFA initial average spin vectors.

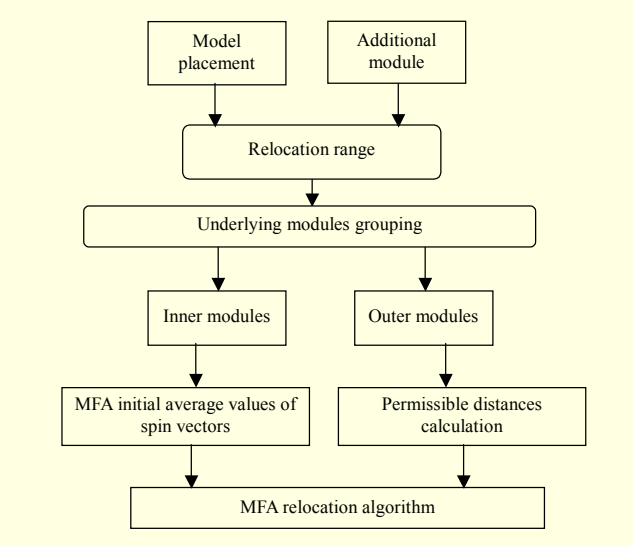

Fig. 4. Flowchart of first stage of local relocation.

represented by

$$\begin{aligned} \text{Horizontal range} &= \left(\frac{\min(x_{in}) - x_{rr}}{Sd}, \frac{\max(x_{in}) - x_{rr}}{Sd}\right), \\ \text{Vertical range} &= \left(\frac{\min(y_{in}) - y_{rr}}{Sd}, \frac{\max(y_{in}) - y_{rr}}{Sd}\right). \end{aligned} \tag{6}$$

The horizontal range is divided into P parts and the vertical range into Q parts. The algorithm then determines the positions of modules based on their vd_i and hd_i values in comparison to P and Q obtained spans.

For module m, being in the i-th vertical span causes the i-th element of v_m^r to become 1, and being in the j-th horizontal span causes the j-th element of v_m^c to be 1. In the MFA space, this means that the probability of finding module m in row i and column j is 1. Here, v^r and v^c are the initial average spin

vectors as two inputs of the MFA algorithm. Figure 4 shows a flowchart of the first stage of the MFA local relocation algorithm.

IV. MFA Relocation Algorithm

The MFA algorithm is derived by analogy to Ising and Potts models, which are used to estimate the state of a system of particles, called spins, in thermal equilibrium [2]-[4]. MFA combines the collective computation property of the Hopfield neural networks [15] with the annealing notion of SA [16].

In the MFA algorithm, discrete variables, called spins (or neurons) are used to encode configurations of combinatorial optimization problems. An energy function written in terms of spins is used to represent the cost function of the problem. Then, with the expected (average) values of these discrete variables, a nondeterministic gradient-descent-type relaxation scheme is used to find a configuration of the spins which minimizes the energy function associated with them [14].

The inputs of the MFA relocation algorithm are row and column permissible-distance matrices, initial vertical and horizontal expected-value vectors of movable modules and some fixed modules, the sizes of modules, and their incident nets.

The proposed MFA algorithm is an optimized form of the MFA algorithm used in [14]. As we will discuss in the energy function section, in addition to common movements, the ability to rotate modules is defined during the relocation process. We will explain that rotations are employed where permissible distances are not maintained. In the section discussing the cooling schedule, we introduce a three-phase cooling process which is a compromise between the cooling process proposed to overcome the placement problem in [14] and that proposed to overcome the relocation problem in [1].

At each stage of the MFA algorithm, one of the movable modules is selected randomly for mean-field-vector calculation from a random selection list, which includes movable modules with unconverged average spin vectors, and then the selected module's average spin vector is updated using this vector. At the end of each stage, the spin of every average vector that is greater than 0.9 is set to 1, and the others are set to 0 and this vector is deleted from the random select list because it has converged.

1. Energy Functions

The MFA algorithm moves modules to minimize the total energy function. Our MFA relocation algorithm's total energy function is the sum of three energy functions. First, the algorithm minimizes the routing cost function or wire length energy, which is the sum of the vertical and horizontal routing costs. Second, the algorithm avoids locating more than one module in same location, thus minimizing the overlap cost. In MFA, the probability of a module being in row i and column j in the same location is computed for all of the modules.

The energy term is formulated corresponding to the overlap cost as [14]

$$\begin{aligned} E_{\mathrm{o}} &= \frac{1}{2}\sum_{i}\sum_{j\neq i}\omega_i\omega_j \\ &\quad \times P\{\text{Modules } i \text{ and } j \text{ are in the same location}\} \\ &= \frac{1}{2}\sum_{i}\sum_{j\neq i}\omega_i\omega_j\sum_{p=1}^{P}\sum_{q=1}^{Q} \\ &\quad \times P\{\text{Module } i \text{ is in location } pq\} \\ &\quad \times P\{\text{Module } j \text{ is in location } pq\} \\ &= \frac{1}{2}\sum_{i}\sum_{j\neq i}\omega_i\omega_j\sum_{p=1}^{P}\sum_{q=1}^{Q} v_{ip}^{r}v_{iq}^{c}v_{jp}^{r}v_{jq}^{c}. \end{aligned} \tag{7}$$

In (7), ω_i and ω_j are the constant weights of modules i and j and are given from a module weight function which is used to encode the areas of modules. These values are input values of the algorithm, and ω_i for module i is related to its area. Here, v_{ip}^{r} is the probability of finding module i in one of the Q locations in row p, and v_{jq}^{c} is the probability of finding module i in one of the P locations at column q.

The third energy function maintains permissible distances.

$$E_{\mathrm{t}} = E_{\mathrm{w}} + a\times E_{\mathrm{o}} + \beta\times E_{\mathrm{pd}}, \tag{8}$$

where E_{t}, E_{w}, E_{o}, and E_{pd} are the total energy function, the routing cost or wire length energy function, the overlap energy function, and the permissible-distances-preservation energy function, respectively. Here, α and β are balance factors between E_{w}, E_{o}, and E_{pd}; α and β are constant during simulation and are used to increase or decrease the importance of each energy function of the total energy function in relation to others.

2. E_{pd} Energy Function and Rotation of Modules in MFA

At each stage of the MFA relocation algorithm, the first temporary maximum value of each module's spin is set to 1, and the others are set to 0. Then, the sums of widths or heights of modules that are placed in same row or column are calculated and the results are compared with the permissible distance of each row or column. If the current location of a module contradicts the permissible distance of its related row or column, rotation is carried out by exchanging the module's width and height. Then, preservation of permissible distances is checked again, and if the problem still exists, the E_{pd} energy function increases as a penalty. Then, the maximum value of

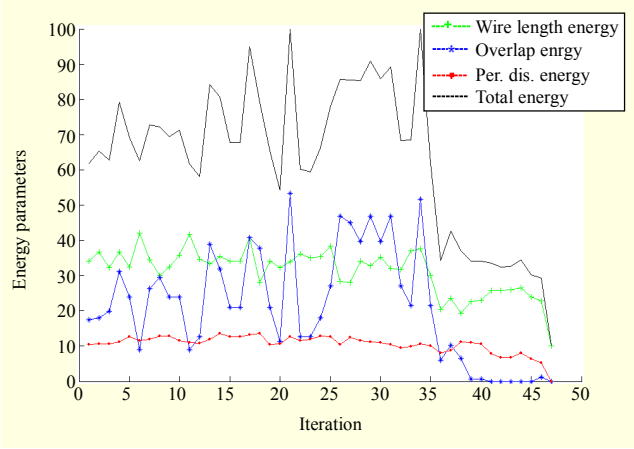

Fig. 5. Energy function minimization: E_t, E_w, E_o, and E_{pd}.

```
if      (0.8×Tr0 ≤ Tr  &  0.8×Tc0  ≤  Tc)
        Tr = 0.95×Tr ; Tc = 0.95×Tc  ;
elseif (0.35×Tr0 <  Tr  <  0.8×Tr0 & 0.35×Tc0  <Tc <  0.8×Tc0)
        Tr = 0.8×Tr ; Tc = 0.8×Tc  ;
elseif (0.35×Tr0 ≥  Tr  &  0.35×Tc0  ≥Tc)
        Tr = 0.65×Tr ; Tc  = 0.65×Tc  ;
end
```

Fig. 6. Cooling schedule algorithm.

the module's spin decreases to about 0. This means that the probability of locating this module in this location is reduced. After updating the average spin values of randomly selected modules, the preservation of permissible distances for all modules is checked again, and a new E_{pd} is obtained.

Change in the E_{pd} value affects the total energy change value. Figure 5 shows the energy function and its parameters, wire length cost, overlap energy, and permissible-distances-preservation energy. In the final stage, in which all of the spins converge, overlap and permissible-distances-preservation energies become 0, and wire-length cost is minimized; therefore, the total energy is also minimized.

3. Cooling Schedule

The cooling process is realized in three phases: slow cooling followed by fast cooling and then very fast cooling.

Figure 6 shows the cooling schedule algorithm. Here, T_{r0}, T_{c0}, T_r, and T_c are the horizontal and vertical initial temperatures and horizontal and vertical current temperatures of the system, respectively.

Because there are vertical and horizontal spins, the two initial temperatures are calculated at the beginning of the algorithm according to the vertical and horizontal dimensions of the problem, and a constant factor is called the initial temperature factor, which is denoted as t_f:

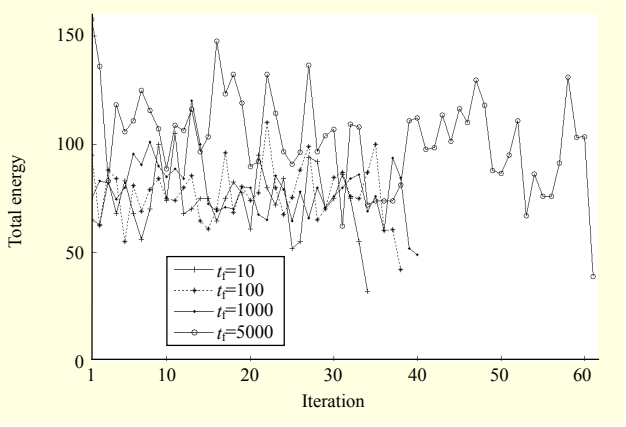

Fig. 7. Total energy function minimization for four values of t_f: 10, 100, 1000, and 5000.

$$T_{r0} \propto t_f \times P, \quad T_{c0} \propto t_f \times Q. \tag{9}$$

Figure 7 shows the total energy minimization during the algorithm iterations for three different values of t_f: 10, 100, and 1,000. It is clear that changing this factor causes the number of iterations and the minimum value of the total energy function to also change.

On the other hand, setting this factor to insufficient values (especially too high values) may cause divergence or unacceptable results; thus, the range of this factor is limited and, according to our experiments, less than 5,000.

The cooling process continues until either 90% of the spins are converged or the temperature is reduced to less than 1% of the initial temperature. In [1] and [14], the cooling process has the same phases.

In [1], the initial temperature is set to 1, and the final temperature is set to 0. This means that when the current temperature falls to 50% of the initial temperature, the task is finished. Our final temperature is 1% of the initial temperature as in [14], so when the current temperature is below 35% of the initial temperature, a very fast cooling phase moderates the unconverged spins.

At the end of this process, the variable with the maximum value in each unconverged spin is set to 1, and all other variables are set to 0. Figure 8 shows the flowchart of the second stage of the MFA local relocation algorithm.

V. Experimental Results

We implemented the proposed algorithm on a 2.4 GHz Intel Pentium IV with 512 MB memory using MATLAB 7.2.0.232 (R2006a) with a Windows operating system. We applied the proposed algorithm to the relocation of n300a, n200a, and n100a, which are distributed according to GSRC benchmarks [17].

For every benchmark, five different problems were resolved

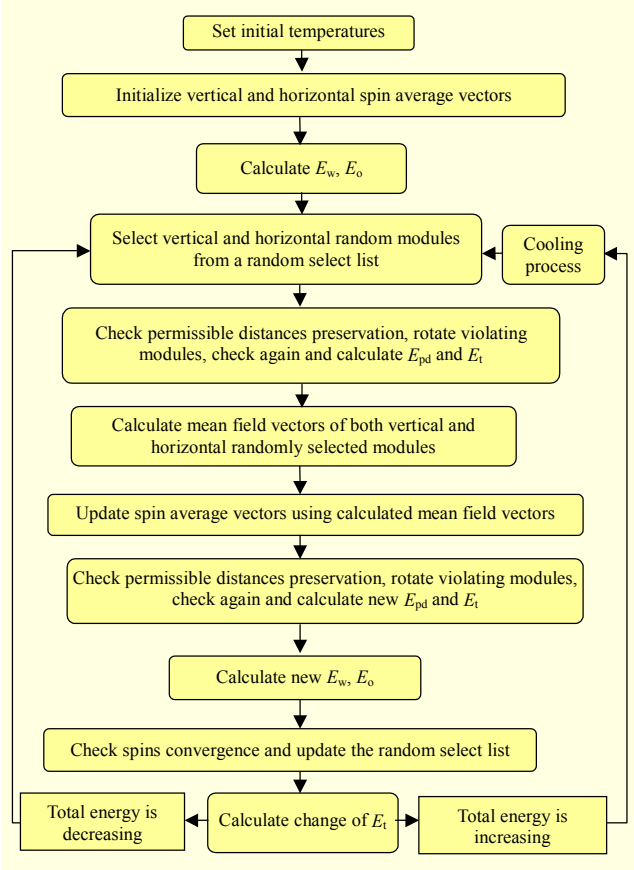

Fig. 8. Flowchart of second stage of MFA local relocation algorithm.

using our proposed algorithm, and maximum and average runtimes of 10 runs are presented in Table 1. The results demonstrate that our MFA-based algorithm is faster than the SA-based method proposed in [1] because the number of displacements is limited to the number of movable modules of the problem, and the problem is local relocation. Actually, the relocation range reflects the number of displacements and the similarity of the resulting placement with the model placement. The results also demonstrate that the runtimes of our proposed algorithm show very little dependence on the size of the benchmark circuit in comparison with the method of [1]. The size of the local relocation range and the numbers of movable modules in each problem are the main parameters here.

Also, the feasibility of the local relocation solution to guarantee the similarity of the resulting placement with the model placement depends on the existence of enough dead space near the additional module so that the relocation range becomes limited and small.

VI. Conclusion

In this paper, a local relocation method was proposed to solve the placement migration problem which results from modifying a model placement by adding an additional module.

Table 1. MFA local relocation results for GSRC benchmarks.

	MFA local relocation			SA
Benchmark	Minimum runtime (s)	Average runtime (s)	Maximum runtime (s)	Minimum runtime (s)
n100	2.0	2.37	2.52	1.0
n200	3.2	3.62	3.72	9.0
n300	3.7	3.92	3.96	60.8

Our proposed method, as a local solution method, causes less displacement than other methods, and by taking advantage of the mean-field-annealing algorithm rather than the simulated-annealing algorithm, it localizes the problem. This reduces the number of engaged modules, and because it has fewer variables, the proposed method is faster. Moreover, having fewer movable modules causes more similarity if the solution is feasible. Selection of modules for relocation is based on the range that includes enough free space around the extra module.

The ability to rotate inner modules, a fixed distance controlling energy function to maintain permissible distances, and a three-phase cooling process are the main properties of the proposed MFA algorithm. The results demonstrate that our method is almost independent of the size and complexity of the model placement.

References

[1] K. Yanagibashi, Y. Takashima, and Y. Nakamura, "A Relocation Method for Circuit Modifications," *IEICE Trans. Fundamentals Elect. Commun. Comput. Sci.*, vol. E90–A, no.12, Dec. 2007, pp. 2743-2751.

[2] D.E. Vanden Bout and T.K. Miller, "Improving the Performance of the Hopfield-Tank Neural Network through Normalization and Annealing," *Biologica Cybern.*, vol. 62, 1989, pp.129-139.

[3] C. Peterson and B. Soderberg, "A New Method for Mapping Optimization Problems onto Neural Networks," *Int. J. Neural Syst.*, vol. 1, no. 1, 1989, pp. 3-22.

[4] L. Gislen, C. Peterson, and B. Soderberg, "Complex Scheduling with Potts Neural Networks," *Neural Computation*, vol. 4, no. 6, 1992, pp. 805-831.

[5] T. Bultan and C. Aykanat, "A New Mapping Heuristic Based on Mean Field Annealing," *J. Parallel Distributed Computing*, vol. 16, no. 4, 1992, pp. 292-305.

[6] M. Ohlsson, C. Peterson, and B. Soderberg, "Neural Networks for Optimization Problems with Inequality Constraints: The Knapsack Problem," *Neural Computation*, vol. 5, no. 2, 1993, pp.

331-339.
[7] M. Ohlsson and H. Pi, "A Study of the Mean Field Approach to Knapsack Problems," *Neural Networks*, vol. 10, no. 2, 1997, pp. 263-271.
[8] J. Hokkinen et al., "A Potts Neuron Approach to Communication Routing," *Neural Computation*, vol. 10, no. 6, 1998, pp. 1587-1599.
[9] L. Herault and J. Niez, "Neural Networks and Graph k-Partitioning," *Complex Syst.*, vol. 3, 1989, pp. 531-575.
[10] D.E. Vanden Bout and T.K. Miller, "Graph Partitioning Using Annealing Neural Networks," *IEEE Trans. Neural Netw.*, vol. 1, no. 2, 1990, pp. 192-203.
[11] R. Cimikowski and P. Shope, "A Neural-Network Algorithm for a Graph Layout Problem," *IEEE Trans. Neural Netw.*, vol. 7, no. 2, 1996, pp. 341-345.
[12] J.S. Yih and P. Mazumder, "A Neural Network Design for Circuit Partitioning," *IEEE Trans. Computer-Aided Design*, vol. 9, 1990, pp. 1265-1271.
[13] T. Bultan and C. Aykanat, "Circuit Partitioning Using Mean Field Annealing," *Neurocomputing*, vol. 8, 1995, pp. 171-194.
[14] C. Aykanat, T. Bultan, and I. Haritaoglu, "A Fast Neural-Network Algorithm for VLSI Cell Placement," *Neural Netw.*, vol. 11, 1998, pp. 1671-1684.
[15] J.J. Hopfield and D.W. Tank, "Neural Computation of Decisions in Optimization Problems," *Biologica Cybern.*, vol. 52, 1985, pp. 141-152.
[16] S. Kirkpatrick, C.D. Gellat, and M.P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, 1983, pp. 671-680.
[17] GSRC Benchmark Suits. Available: http://vlsicad.eecs.umich.edu/BK/CompaSS

Gholam Reza Karimi received the BS, MS, and PhD in electrical engineering from Iran University of Science and Technology (IUST) in 1999, 2001, and 2006, respectively. Since 2007, he has been an assistant professor in the Electrical Engineering Department at Razi University, Kermanshah, Iran. His research interests include low power analog and digital IC design, radio-frequency IC design, and modeling and simulation of mixed signal IC.

Ahmad Azizi Verki received the BS and the MS in electrical engineering from the Islamic Azad University of Qazvin and the Razi University of Kermanshah, in 2006 and 2009, respectively. His research interests include VLSI circuit design algorithms.

Sattar Mirzakuchaki received the BS in electrical engineering from the University of Mississippi in 1989, and the MS and PhD in electrical engineering from the University of Missouri-Columbia, in 1991 and 1996, respectively. He has been a faculty member of the College of Electrical Engineering at the Iran University of Science and Technology, Tehran, since 1996. His current research interests include characterization of semiconductor devices and design of VLSI circuits. Dr. Mirzakuchaki is a member of IEEE and IET (formerly IEE) and a Chartered Engineer.