Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/090343

Multi Objective Particle Swarm Optimization based Mixed Size Module Placement in VLSI Circuit Design

Gholamreza Karimi[*], *Hosna Akbarpour and Arash Sadeghzadeh*

Electrical and Electronics Engineering Department, Razi University, Kermanshah-67149, Iran

Received: 13 Aug. 2014, Revised: 14 Nov. 2014, Accepted: 15 Nov. 2014
Published online: 1 May 2015

Abstract: Placement process is one of the vital stages in physical design. In this stage, modules and elements of circuit are placed in distinct locations according to optimization basis. Placement algorithms try to minimize the longest delay along the paths in the circuit and/or minimize the total wire length. So placement is an important step in circuit architecture. After it, we can reach to routing stage. It is known that particle swarm optimization (PSO) is one of the practical evolutionary algorithms for this kind of applications. In this project, a novel method for optimized module placement has been used. According to this process objectives in this issue were wirelength and overlap removal function, consequently we were forced to use multi-objective particle swarm optimization (MOPSO) in the algorithm. Structure of MOPSO is in a way that introduces set of answers, we had been tried to find a unique answer with minimum overlap. This algorithm has less run time as compared with other methods of placement, run time plays an important role in VLSI circuit design. Also the experiments on GSRC benchmarks show that the proposed algorithm is effective, and gives out many optional results for users choice in the physical design of VLSI circuits.

Keywords: MOPSO, Optimization algorithm, Placement, VLSI design, wirelength cost function, overlap removal

1 Introduction

Placement is one of the important steps in circuit design. In this stage, modules and elements are placed in distinct locations. In total design process of circuit, this stage consumes most time. So algorithms with fast response and better convergence are expected for meeting requirements. In modern layouts, increasing in transistor numbers and decreasing size of elements make us let the die surface smaller and smaller, meaning that we must minimize involved space of modules. Smaller die size results in needing more complex and precise algorithms. Placement is a stage in physical design that maps layout structure in surface of the chips.
Placement problem is considered as NP-complete problem [1]; meaning that it is a difficult problem in computation aspect. So, a unique answer can not be dedicated to it, and algorithms are used to get a set of answers for solving design necessities. Structure of applied algorithms is heuristic nature.
The VLSI placement becomes much more complicated when circuit contains modules of different sizes. So problem definition is as follows:
Inputs of problem are:

– Set of modules
– Set of nets

Outputs of problems are:

– Optimized layout of circuits

Goal of process is:

– Optimizing various kinds of parameters like wire length, wire congestion, signal delay, power, performance etc.

Optimization of cell placement algorithms can be divided into following categories:

1. Partitioning based algorithms
   Many of placement algorithms can be sorted in this part. Each portion of problem divides into some sub portions, and optimization algorithm is applied in each one separately. In [2] a method based on division and replacement was introduced that focuses on getting better values of wire length.
2. Analytic algorithms
   In this category, optimization techniques are based on, merely, computational procedure. Cost functions are

[*] Corresponding author e-mail: ghkarimi@razi.ac.ir

expressed based on mathematical formula and optimization is done according to this method to get proper values. Models like methods in [3] are classified in this list. In this method, the main consideration that modules must have no overlap with each other is important. Also according to pre-defined hypergraph structure, set of answers in mathematical format is introduced.

3.Evolutionary algorithms
These kinds of algorithms mostly offer random answers during optimization process; in this category, one location of modules may differ from other iterations. One of the most applicable algorithms in cell placement is SA, lays in this category. The suggested model used in [4] explains principles of this algorithm. In [5] MFA algorithm is brought up.

Better run times is sought in locations called *frame* , a distinct chosen space from the whole of chip's surface, that algorithm is applied on. Our work is similar to method introduced in [6], we first select a surface called frame, that involves some modules. Then using PSO algorithm accomplishes our target about wire length and overlap removal. In this article, we compare PSO with two other VLSI placement algorithms. Simulated annealing (SA) is one of the most applicable algorithms in placement issue. In fact, this algorithm simulates the problem to the annealing process (melting metal and slowly cooling it). Module placement is done beside a cooling process; it starts with a random solution. Algorithm consists of number of movements that leads to change in temperature and its corresponding parameters. Each movement can be accepted if cost decreases, this process goes on until a stable state with reasonable cost value to be gained. But the main drawback of it is its greatly run-time, and we need laborious cost functions to have a proper placement.
The other algorithm is MFA or mean field annealing, based on neural network basis and again cooling process, and much faster. In this structure, the concept of hypergraph is used and theory goes on by dividing the surface of chip to equal squares. We tried to work on a new algorithm with fast convergence rate that can optimize our conflicting objectives simultaneously.
But why we decide to work on wire length and overlap? Wire length has direct effect on power consumption. The smaller wire length cost, the more economical circuits.
In the other hand, overlaps make our chips design full of fabrication and routing obstacles, so its optimization is mandatory.
The remaining of paper is as follows:
In section 2, we explain particle swarm optimization (PSO)'s theory, history and principles. Its concept of optimization is discussed briefly. Then multi-objective PSO (MOPSO) is introduced and explained. Section 3 is dedicated to cost function. Wire length structure and overlap issue are points of survey in this section. Experimental results and benchmark tests are shown in section 4. At last, we have conclusion in section 5.

2 Optimization algorithm

2.1 Principles of algorithm

This algorithm of optimization was introduced by Kennedy and Eberhart in 1995. A swarm in PSO consists of number of particles. Each particle represents a potential solution of optimization task [7].PSO acts according to swarming theory and is inspired of social behavior of some animals, like bird flocking and fish schooling. In fact, this method is a population based method.Each particle in PSO has its own position and velocity, finding a better answer encourages position and velocity to change value toward that. So, adequate iteration can make a good answer.
Beside other evolutionary algorithms, this is a simpler one and the rate of convergence of it is faster. We start algorithms by set of random answers and search is done in parallel to get best answers. Structure of each particle is influenced by 2 factors:

–Best state that particle has been achieved or *pbest*
–Best state that is achieved by all of the particles or *gbest*

Algorithm uses concepts of velocity and position, new position of each particle is obtained from previous velocity and position.

2.2 PSO

For each particle i, we have position and velocity vectors as:

$$x_i = [x_{i1}, x_{i2}, .x_{in}] \tag{1}$$

$$v_i = [v_{i1}, v_{i2}, .., v_{in}] \tag{2}$$

Where n is number of decision parameters of an optimal problem. And we have:
pid=position of previous *pbest*
gid=position of previous *gbest*
And we assume that *xid(t)* and *vid(t)* are position and velocity of i-th particle in t-th iteration. By all of these considerations we have:

$$v_{in} = wv_{in} + c_1 r_1 (p_{id} - x_{in}) + c_2 r_2 (g_{id} - x_{in}) \tag{3}$$

$$x_{in} = x_{in} + v_{in} \tag{4}$$

Where w is inertia weight of velocity in the range of [0,1] and *c1*,*c2* named acceleration coefficients, also *r1* and *r2* are two random numbers that are uniformly generated between 0 and 1.

First term of equation 3 is called inertia and considers the current state of particle. Second term or cognitive term, shows distance from best state in neighborhood. The third term or social learning term shows distance from best answers in entire search space. If the sum of these three values exceeds from maximum value defined for velocity (*Vmax*), *vid* should be equal to *Vmax*.
Algorithm with bigger *Vmax* has bigger steps in search space and considers far points, while smaller *Vmax* has the potential of local optimizations.Based on these expressions, the Pseudo code of standard PSO algorithm is as follows:

```
Initialize positions and velocities of all particles in the swarm randomly
Repeat
For each particle in the swarm
Calculate the fitness value f(xi)
If f(xi) >f(pid) then pid=xi
End for
Update Pg, if the best particle in the current swarm has lower f(x) than f(gid)
For each particle in the swarm
r1=rand (); r2=rand ();
Calculate particle velocity according to equation 2
Restrict the velocity of particles by [-(Vmax), (Vmax)]
Update particles position according to equation 2
End for until maximum iteration or a minimum error criterion is attained.
```

2.3 MOPSO

The successful application of PSO in many single objective problems reflects its effectiveness, and it seems to be suitable for multi-objective optimization due to its efficiency in yielding better quality solutions while requiring less run time. In optimization problems with multi-cost function, we must set a trade-off between objectives. Sometimes in two objective problems, both objectives may be in conflict and compete with each other. In this situation, we can't find a unique answer for the problem, but often we have a set of answers that logically can optimize cost functions, this set of answer is known as Pareto answers. We are going to minimize a function defined as:

$$f(x) = f_1(x), f_2(x), ., f_m(x) \qquad (5)$$

Where m is number of the objectives and D is feasible search space [8]. x can be a member of D The algorithm should optimize f(x) and produces Pareto solution, Pareto is a set of non-dominated solutions.
If no objective can be improved without sacrificing other objectives, we must set a trade-off. The main difficulty in extending PSO to multi-objective problem is to find the best way for selecting the guides of particles in that intended swarm, the difficulty is manifest as there are no clear concepts of personal and global bests that can be clearly identified when dealing with objectives rather than a single objective [9]. For classification of answer set, we use an abstract space called repository. Members of repository are our answers.
Steps of an ordinary MOPSO algorithm are:

1.Initialize first population
2.Find non-dominated members and put them in repository
3.Grid the search space
4.Choose leader for each particle from repository members
5.Update best state
6.Add new non-dominated members to repository
7.Delete dominated members of repository
8.Check size of repository and compare in with max size

3 Cost function

Solving the problem makes us to introduce wire length and overlap as cost functions of MOPSO theory. Because there is no exact answer that can optimize both cost functions, we are forced to consider a set of answers.
Attention to random access of the algorithm, often it is needed to run it for many times, to achieve one reasonable answer. We assume overlap is much more important than wire length, because if two modules have overlap with each other, optimized wire length can't lead to a proper structure mapped on a chip. So between repository members, we select answer with minimum value of overlap. It is seen according to PSO algorithm, if we consider only wire length as cost function, locating modules for minimizing turns on plenty of overlaps. So it is clear that two cost functions must be optimized with respect to each other. Two cost functions are described as:

3.1 Wirelength

One of the major parameters based on optimization structure is wire length. This factor also is one of the important elements in circuit designs. Both wire length and wire congestion are types of optimization goals in placement issue, but running fully optimization rules for these two factors is unattainable. So, for having good optimization manner, there must be trade-off between them. In this work, we concentrate on the wire length as first member of the cost function vector. In a set of modules, lay in surface of a chip, wiring is done between common-net modules. This value is multiplied in weighting coefficient (*wnet*). The left-down point of each module is considered as introduction point of it as we have in figure1:

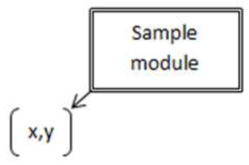


Fig. 1: structure of a single module

Diverse methods are reported for calculating wire length in FPGAs. For example in [10] distance between two points can be written as:

$$Wirelength = |x_i - x_j| + |y_i - y_j| \qquad (6)$$

The wire length calculated as the length of the bounding box of each wire. In [11] , cost function was computed by the sum of wire length values. All above methods give us wire length in FPGAs. By generalization this procedure to mixed-size modules, we will have pseudo code of wire length issue as:

```
Wtot=0
For each row of matrix net
Wtot1=0
For each two separated columns of net
Calculate x,y of each two columns
$Wtot1 = Wtot1 + |x1 - x2| + |y1 - y2|$
Wtot1=wnet*Wtot1
Wtot=Wtot+wtot1
```

So wire length can be considered as a row of total cost function vector.

3.2 overlap

Undoubtedly, one of the important and basic goals in placement process is that modules have no overlap with each other. Overlap causes fabrication problems, routing problems and many others. Overlap function, as a cost function, seeks in search spaces in a way to remove overlap between modules by iterations. A single module in figure 2 is defined as:

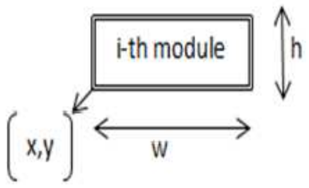


Fig. 2: module with its corresponding sizes

We have various *w* and *h* values derived from this fact that our modules are mixed-size. This object makes overlap survey more complicated.
If we depict inner module by blue color and overlapped modules by orange color, state of overlaps are:
First we consider left side of inner module, 4 states of overlap can be shown. Overlap of left sided module is shown in Figure 3(a). For right sided part of module, we have Figure 3(b). And central overlap of inner modules is illustrated in Figure 3(c), at last, 2 states are devoted to inside and outside overlaps, as demonstrated in Figure 3(d)

Each of existing overlaps produces a factor of penalty. In overlap removal process, minimizing the penalty factor is followed (when we have no overlap, Minimum value is accessed). Based on initial location of modules, (x,y) that is assigned by PSO and corresponding values of w and h, we can show overlaps. In this algorithm, we try to look for answers of

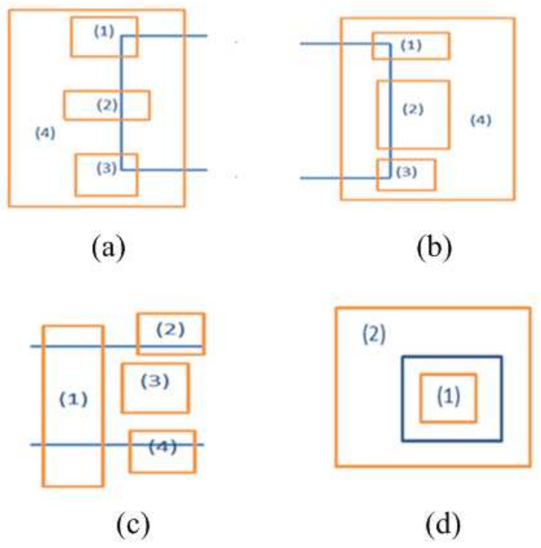


Fig. 3: overlap of modules (a):overlap of left sided module, (b):overlap of right sided module, (c):overlap of center sided module, and (d): inner and outer overlap

placement or *relocation* problem, involved wire length and overlap optimizations. A set of answers, called *repository* and concepts like *cost* and *best cost* in the MOPSO structure are defined.
Repository has limited capacity and only non-dominated answers can enter it. For each of iterations, it is updated. Repository members are answers of MOPSO algorithm (Pareto answers). In our problem, cost functions are wire length and overlap that a trade off is needed between them. Minimum overlap cost value states are chosen and compared with other answers to find minimum wire length, so from a set of answers we can get a single one.

4 Experimental results

According to the discussed issue, this PSO algorithm can be applied to set of data. We implemented the proposed algorithm on an Intel core i7 with 8GB memory using MATLAB (R2009a) with windows operating system. In each of sample data, a space that has much free locations was selected. Then modules that were involved in selected lines were considered as frame of other inner modules inside selected space. Inner modules were inserted in algorithm until good values for answer had gotten. By iterations states with no overlap, reasonable wire length could be reached.
We applied the algorithm to the relocation of n100, n200 and n300 which are distributed according to GSRC benchmark [12]. As a sample of our results, n100 modules algorithms structure is represented in figure 4, fixed modules and inner modules are shown in two different colors, modules surrounded in dashed line are inner modules, modules overlapped by mentioned line are frame, and other modules are outer.

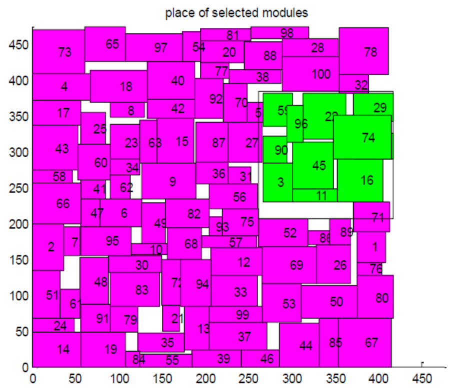


Fig. 4: module location for 100number test data

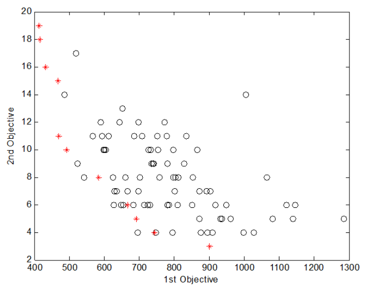
Results for n100, n200 and n300 modules are summarized in following Table (Table 1), as comparison with SA and MFA run times, we have:

Table 1: comparison of different placement methods run-time

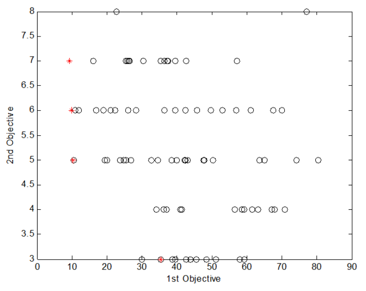
run time	*Min(s)*	*Max(s)*	*Average(s)*	*SA(s)*	*MFA(s)*
n100	0.702	1.2324	0.8776	1.0	2.37
n200	0.7176	1.1232	0.8296	9.0	3.62
n300	1.5147	1.7784	1.6208	60.8	3.92

The results demonstrate that our MOPSO-based algorithm is faster than MFA, because naturally PSO algorithm has very fast convergence rate in comparison to other algorithms. And faster than SA because the number of displacements is limited to the number of movable modules of problem and the problem is local relocation. The results also show that run times in this problem are almost independence on the size of the benchmark circuit. Using reasonable iterations, a good result can be obtained. Figure 5 shows cost functions (wire length and overlap removal) of n100, n200 and n300 benchmarks.

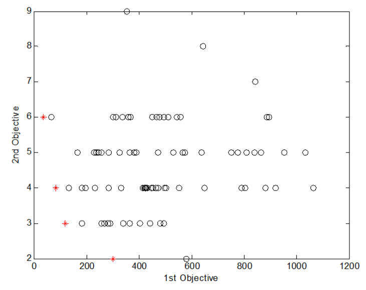
The spots in mentioned plots are considered PSO particles in last update and best results in one algorithm process. As shown, only limited numbers of particles can be laid in repository. In fact, just these few numbers can meet our requirement and optimization goals.
First objective is devoted to wire length cost function and is the calculation of wires of inner modules (modules inside the frame), so each of mentioned figures may have different values of its objective from others. Times of overlap occurrence also are showed in second objective. Minimum overlap case is selected between members in lowest level of vertical axis (overlap axis).
Drawing of these figures is completely random, means each time we repeat the algorithm, we get a new one. These figures consist of white and red points, red points are repository members. Final answer has been chose between repository members.
Figure 6 shows wire length and overlap cost functions according to iterations in the algorithm process. Concentrating on trade off



(a)



(b)



(c)

Fig. 5: graph of objectives for (a): n100 modules, (b):n200 modules and (c):n300 modules

issue and proper comparisons forced us to divide the real wire length to a constant coefficient (sometimes different in these three benchmarks).
As mentioned, repository is a set with the most possible answer of problem. These plots show mean values of normalized wire length and overlap for comparison.
By completing the algorithm, values of wire length and overlap cost functions decrease. In the other hand, the relative contrast of the cost functions can be perceived, meaning where we have low wire length cost, overlap cost nearly increases and vice versa.
The art of using MOPSO algorithm is that to propel the results to decreasing two cost functions, and choose the best answer according to optimization benefits.

Particle costs and repository costs are in normalized wire length and overlap aspects.
Figure 7 shows repository members and best results of ordinary particles in the field of comparison. According to this plots, we can prove that the cost of repository members usually is smaller

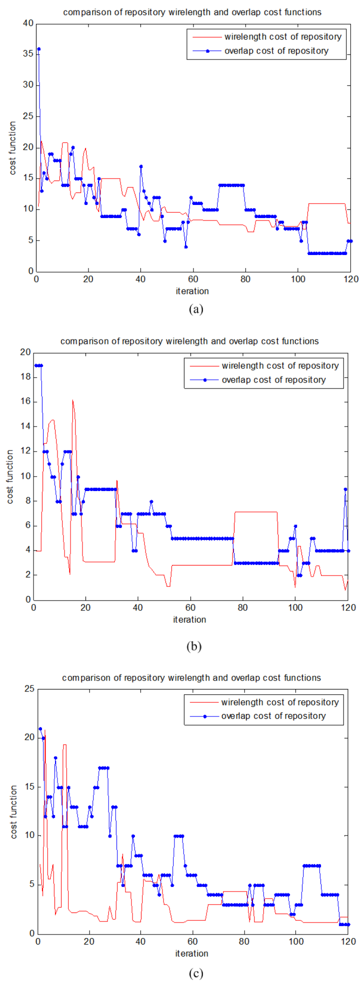


Fig. 6: Comparison of repository cost of wire length and overlap in (a):n100 benchmark, (b):n200 benchmark and (c): n300 benchmark

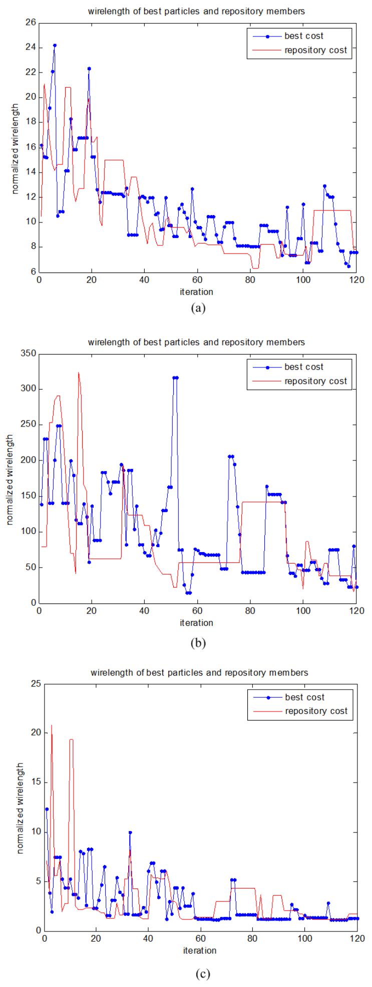


Fig. 7: Comparison of best cost and repository cost of wire length in (a):n100 benchmark, (b):n200 benchmark, and (c): n300 benchmark

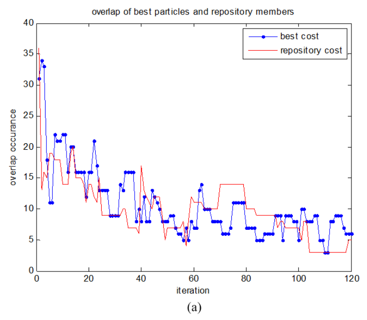
than cost of ordinary particles, but this point must be expressed that in comparison. Therefore the contrast between wire length and overlap values should be considered.

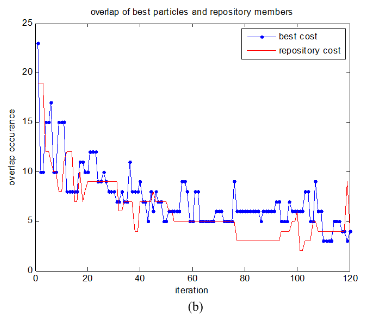
As mentioned, wire length is divided to a constant coefficient, and for overlap we have in figure 8:

It is necessary to mention that increasing in number of modules (for example 300module), makes calculations more and more complicated. And finding answers with no overlap and proper wire length becomes a very complex problem.

5 Conclusion

In this paper, we introduced a multi-objective PSO (MOPSO) algorithm for placement stage in VLSI circuit design. It is known that placement is one of the most important and applicable issues in manufacturing process and reaching to optimized results can be done by various procedures. Iteration methods and evolutionary principles got us to an algorithm that optimized answer of designs by relocating the modules. Simple PSO with its fast convergence rate and high speed attracts many researchers of optimization fields. But various cost functions usually need a new version of used PSO algorithm. Two considered cost functions in this work are wire length and overlap removal. During the process, sometimes, we were forced to sacrifice one objective to another one, for selecting no-overlap solution. As comparison with other known placement algorithms, better run times were gained, beside overlap occurrence and total wire length were optimized properly, and run time almost was independent of number of modules. Data were tested by standard benchmarks and was compared to previous methods.





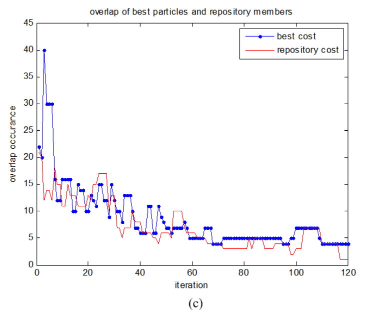


Fig. 8: Comparison of best cost and repository cost of overlap in (a):n100 benchmark, (b): n200 benchmark, and (c): n300 benchmark

References

[1] B.Sekhara, R.Swethar, S.DeviK.A. Comparison of Hierarchical Mixed-size Placement Algorithms for VLSI Physical Synthesis, International conference on communication systems and network technologies.430-435,(2011).

[2] Y.Takashima, A.Kaneko, S.Sato and M.Kaneko. Two-Dimensional Placement Method Based on Divide-and-Replacement, Conference on Circuits and Systems.vol.2, 341-346,(2002).

[3] Z.Jiang, H.Chen, T.Chen and Y.Chang, Challenges and Solutions in Modern VLSI Placement, International symposium on VLSI Design, Automation and test,1-5,(2007).

[4] K.Yanagibashi, Y.Takashima, and Y.Nakamura. ARelocation Method for Circuit Modifications, IEICETrance.Fundamentals,Elect.commun.comput.sci. vol.E90-A, no.12,2743-2751,(2007).

[5] C.Aykanat, T.Bultan and I.Haritaoglu. A Fast Neural Network Algorithm for VLSI Cell Placement, Neural Netw.vol.11,1671-1684,(1998).

[6] G.Karimi, A.Aziziverki and S.Mirzakuchaki.Optimized Local Relocation for VLSI Circuit Modification Using Mean-Field Annealing, Electronic and Telecommunications Research Institute(ETRI) journal. vol.32, no.6, 932-939,(2010).

[7] S.Hsieh, C.Lin, T.Sun. Particle Swarm Optimization for MacrocellOverlap Removal and Placement, Swarm intelligence symposium.177-180,(2005).

[8] M.Reddy and N.Kumar. Multi-Objective Particle Swarm Optimization for Generating Optimal Trade-offs in Reservoir Operation. Hydrological processes, vol.21,2897-2909,(2007).

[9] J.Alvarez-Benitez. R.Everson and J.Feildsend,A MOPSO Algorithm Based Exclusively on Pareto Dominance Concepts, department of computer science, university of Exeter, UK.459-473,(2005).

[10] M.EL-Abd, H.Hassan and M.S.Kamel. Discrete and Continuous Particle Swarm Optimization for FPGA Placement, IEEE Congress on Evolutionary Computation.706-711,(2009).

[11] P.K.Rout, D.P.Acharya, G.Panda. Digital Circuit Placement in FPGA Based on Efficient Particle Swarm Optimization Techniques,5th international conference on industrial and information systems. 224-227,(2010).

[12] GSRC Benchmark Suits, Available: http://vlsicad.eecs.umich.edu/BK/CompaSS

Gholamreza Karimi received the B.S. and M.S. and PhD degrees in electrical engineering from Iran University of Science and Technology (IUST) in 1999, 2001 and 2006 respectively. He is currently an Assistant Professor in Electrical Department at Razi University, Kermanshah, since 2007. His research interests include low power analog and digital IC design, RF IC design, modeling and simulation of RF mixed signal IC, microwave devices and artificial intelligence systems.

Hosna Akbarpour received the BS and the MS in electrical engineering from the Razi University of Kermanshah, in 2009 and 2013,respectively. Her research interests include VLSI circuit design algorithms, PSO structures like Multi objective and Discrete versions.

Arash Sadeghzadeh was born in Shiraz, Iran, in 1979. He attended the high school run by NODET (National Organization for Developing Exceptional Talents) and received his B.Sc. in Biomedical engineering from Amir Kabir University (Tehran Polytechnic). He obtained the M.Sc. degree in control engineering from Tehran University in 2003 and the Ph.D. degree in Automatic Control from Tarbiat Modares University in 2010. He is currently an assistant professor at Razi University, Iran. His main research interests include robust control and identification for control.