Design and Analysis of Multi-level Genetic Algorithm with its Application to the Construction of Clock Binary Tree

Nan Guofang, Li Minqiang, Kou Jisong

Institute of Systems Engineering, Tianjin University, 300072, China

Summary
Genetic algorithm is an effective methodology for solving combinatorial optimization problems, and numerous researchers have undertaken efforts to many kinds of improvement of GAs in order to solve problems in computer science. The clock signal and clock skew become more and more important for the circuit performance in VLSI layout design. Since there are salient shortcomings in the conventional topology construction algorithms for designing a clock network, the multi-level model of clock binary tree is built in this paper, and the binary tree construction algorithm of clock signal based on multi-level genetic algorithm (MLGA) is presented. The experiments on random test cases and standard benchmark test cases show that multi-level genetic algorithm can produce much better clock network design in most cases when compared with conventional heuristic algorithms.

Key words:
Clock Skew, Genetic Algorithm, NP hard, Partial Mapped Crossover .

1. Introduction

In synchronous systems, the circuit performance is directly proportional to its clock frequency. A clock net needs to be routed with great precision, since the actual length of the net path from its source point to sink points determines the maximum clock frequency on which a chip may operate. In addition, the clock signal must arrive simultaneously at all functional units with little or no waveform distortion [1]. With the development of VLSI technology, the circuit speed is limited more and more rigidly by two factors: 1) delay on the longest path through combinational logic, 2) clock skew, which is the maximum difference in arrival times of the clock signal at the units [2]. This is shown from the following inequality.

$$t_p \geq t_d + t_{skew} + t_{su} + t_{ds} \tag{1}$$

Where t_p is the clock period, t_d is the delay on the longest path through combinational logic, t_{skew} is the clock skew, t_{su} is the set-up time of the synchronizing elements, t_{ds} is the propagation delay within the synchronizing elements. Increased switching speeds due to advances in VLSI fabrication technology will significantly decrease the terms t_d , t_{su} , and t_{ds} . Therefore t_{skew} becomes the dominant factor in determining circuit performance [2].

2. Problem Formulation

We call the positions for elements of a circuit the sinks of the clock net. A set of sink locations, which is usually denoted by $S = \{s_1, s_2, ..., s_n\} \subset \Re^2$, specifies an instance of the clock routing problem. A connection topology is defined to be a binary tree G , which has n leaves corresponding to the set of sinks [2]. $T_G(S)$ is an embedding of the connection which associates a placement with each node $v \in G$, and $l(v)$ represents the location. The root of the clock tree is the clock source. e_v is the connection edge between the node v and its father node, and the value of e_v , denoted by $|e_v|$, represents the real pathlength. The total pathlength of the clock tree, denoted by $\cos t(T)$, represents the sum of all edges. $t(u,v)$ is the signal delay from node u to node v .

$$skew(T) = \max_{s_i, s_j \in S} |t(s_0, s_i) - t(s_0, s_j)| \tag{2}$$

Theoretically, the algorithm discussed in this paper is a zero skew method, so the objective function is

$$\cos t(T) = \sum_{e_v \in T} |e_v| \tag{3}$$

In the object function, the linear pathlength delay model [3] is adopted to calculate the signal delay from clock source to each clock sink.

The first clock tree construction algorithm is H-tree method, which is used in regular arrays. The MMM (Methods of Means and Medians) algorithm [4] generates a topology by recursively partitioning the set of sinks into two equal-sized subsets, and then connects the center of the subsets. A bottom-up matching approach to clock tree construction, proposed by Kahng [5], can eliminate all source-sink pathlength skews. The deferred merge

Manuscript received February 25, 2006.
Manuscript revised February 28 , 2006.

embedding (DME) method [6], proposed by three independent groups, is a linear-time algorithm, and is a comparatively successful algorithm used in clock routing. Some improvement algorithms [7-8] are derived from the DME method and other technologies including buffer insertion, Elmore delay model, exact zero-skew RC model, and so on. But all of these are heuristic methods, we tend to apply a global optimization algorithm to clock routing problem.

We present the multi-level model of the rooted binary tree, and design the corresponding multi-level genetic algorithms (MLGA) in Section 3. Numerical experiments on test problems are implemented in Section 4. In Section 5, we conclude the paper.

3. Multi-level Genetic Algorithm

GA [9-11] is an adaptive heuristic search algorithm premised on the evolutionary ideas of natural selection. GAs have been widely studied, experimented and applied in many fields in engineering worlds. Not only does the GA provide an alternative method to solving problem, it outperforms other traditional methods on many hard optimization problems. In the clock routing problem, when the clock source and all clock sinks are given, the structuring of the rooted binary tree with minimum cost is the main task. A novel style of GA is proposed to solve this problem by employing specific encoding scheme and genetic operators.

3.1 A Simple Genetic Algorithm

Genetic algorithms are a class of optimization algorithms that seek improve performance by sampling areas of the parameter space that have a high probability for leading to good solutions [12]. The algorithms are called genetic because the manipulation of possible solutions resembles the mechanics of natural selection.

As an optimization technique, genetic algorithms simultaneously examine and manipulate a set of possible solutions [13]. Each candidate solution is represented by a string of symbols called chromosome, a chromosome of length n is defined to be $(s_1, s_2, ..., s_n)$. A genetic algorithm starts with an initial population, which consists of a set of solutions. This population then evolves into different populations for a large number of iterations. At last, the algorithm returns the best individual as the final solution to the problem. For each iteration or generation, the evolution process proceeds as follows [14]. Two members (parents) of the population are selected based on some probability distribution. These two parents are then combined through a crossover operator to produce an offspring. With a low probability the offspring is modified by a mutation operator to introduce an unexplored search space to the population, the diversity of the population is then enhanced. In this way, a number of offspring are generated, and they replace part of the whole population. We now have a new population, and the evolution process is repeated until a certain condition is met, for example, after a fixed number of generations. The flowchart of genetic algorithm [15-18] is shown below.

```
Genetic algorithm
1. encode solution space
2. set pop_size, max_gen,gen=0;
  set crossrate, mutationrate;
3. initialize population
4. while max_gen>=gen
        evaluate fitness
        for(i=1 to popsize)
          select(mate1,mate2)
          if (rand(0,1)<crossrate,
          child=crossover(mate1,mate2);
          if(rand(0,1)<mutationrate,
          child=mutation(chromosome);
        end for
        add offspring to new generation.
        gen=gen+1
      end while
5. return best chromosomes
```

3.2 Principles

Given the clock source s_0 with its coordinate (x_0, y_0), and all clock sinks $s_i (1 \le i \le n)$ with their coordinate (x_i, y_i), we intent to construct a topology including source and sinks as a rooted binary tree such that the total pathlength is minimized. In this paper, the bypass and confliction of internal nodes will not be considered, and the real distance between different nodes is used to represent their routing pathlength. According to characteristics of a binary tree, we divide it into multiple levels, and every level is treated respectively from the bottom (leaf nodes) to the top (root nodes), which obeys the principles as below.

(a) The node sequence in every level is determined according to basic operation of genetic algorithms, two adjacent nodes are lined in sequence, and the merging point is decided by a heuristic merging scheme, therefore, the topology of this level and some internal nodes of the binary tree are achieved.

(b) Any level's output is treated as the input of its next level, that is, all the internal nodes got in one level are leaf nodes of the next level. The GA is executed to get new topology and new internal nodes in a level.

(c) The step (b) is executed repeatedly until the root (clock source) of a binary tree is reached.

(d) The objective function of a level is the total pathlength, the reciprocal of which is used as the fitness function.

(e) The merging scheme is a zero-skew scheme.

Based on above procedures, a multi-level genetic algorithm frame with its corresponding fitness function and genetic operators are presented as the following sections.

3.3 Algorithm Description

A clock binary tree is constructed through a multi-level procedure based on GA, which adopts similar genetic strategies including encoding, selection, crossover, mutation and fitness function in each level. The control parameters of the GA don't remain the same, and they vary from the bottom level to the top level.

A. Encoding

The MLGA is a bottom-up approach to a clock tree construction. Considering the first level, the input is n clock sinks $s_1, s_2, ..., s_n$ and their serial numbers are $1,2,...,n$, so that the integer encoding can be used in binary tree construction for the MLGA. A chromosome $\{ a_1, a_2, ..., a_i, ..., a_n \}$ $(1 \le a_i \le n, 1 \le i \le n)$ denotes an individual which is randomly generated but differs in value with each other. The problem solution space is consisted of chromosome strings.

According to the characteristics of forming a binary tree, two adjacent nodes should be lined in sequence and one node is used only once (as shown in Fig 1). Through running a GA, we get the best individual, and the merging of points can be achieved by implementing the merging scheme. There are $n = 2k$ $(k = 1,2,3,...)$ nodes in the first level, k merging points (internal nodes) can be achieved, which are treated as input nodes for the next level. Repeatedly execute the GA until we get to the root (clock source) of a binary tree.

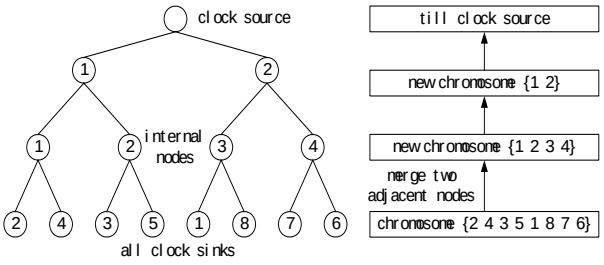


Fig 1. Encoding of constructing a binary tree

B. Fitness

In the multi-level model, every level's pathlength should be minimized separately, and the fitness for an individual is the reciprocal of the objective function value.

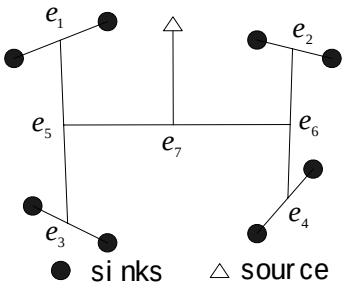


Fig 2. A bottom-up binary tree

A binary tree may be divided into m levels, as shown in Figure 2. $\{e_1, e_2, e_3, e_4\}$ is the corresponding pathlength set to form the first level, $\{e_5, e_6\}$ is the one to form the second level, and $\{e_7\}$ is the one to form the third level. In the i th level, r lines with its length $|e_j|$ $(1 \le j \le r)$ are summed up as the total pathlength:

$$l_i = \sum_{j=1}^{r} |e_j| \quad (1 \le j \le r) \quad (1 \le i \le m) \tag{4}$$

Therefore, the fitness function in the i th level can be defined as:

$$f = \frac{h}{l_i} = \frac{h}{\sum_{j=1}^{r} |e_j|} \quad (1 \le j \le r) \quad (1 \le i \le m) \tag{5}$$

Where h is a constant parameter for a specific clock network, we take $h = 100$ in this paper

C. Selection

In order to ensure proper selection pressure, the roulette wheel selection strategy is used. The roulette wheel selection is a proportionate selection scheme in which the slots of a roulette wheel are sized according to the fitness of each individual in the population. The probability of selecting an individual is therefore proportional to its fitness. Meanwhile, the elitist strategy is used to retain the best individual directly to the next generation.

D. Crossover

There are three styles of crossover operators usually used in applications as order crossover, partially mapped crossover and cycle crossover. Here, the partial mapped

crossover (PMX) is adopted for MLGA. It is implemented as follows. When a random cut point is chosen, the segments following the cut point in both parents are considered as partial mappings of the modules to be exchanged. We take corresponding modules from the segments of both parents, and locate both these modules in the first parent and exchange them. This process is repeated for all modules in the segment. Thus a module in the segment of the first parent and a module in the same location in the second parent will define which modules in the first parent have to be exchanged to generate the offspring. An example is shown in Fig 3.

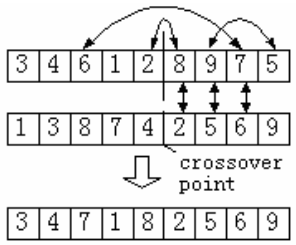


Fig 3. PMX crossover

E. Mutation

The exchange mutation is often used for the integer encoding in a GA. When two bits are randomly selected from an individual to be operated, an exchange between their corresponding nodes will be made. But in clock routing problem, random exchange will cause invalid operation possibly as shown in Fig 4. If two modules (2,4) to be exchanged are in the same block, there is no sense in the offspring chromosome.

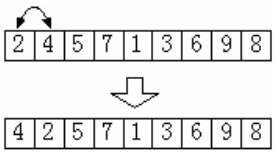


Fig 4. An example for mutation

Therefore, we adopt improved exchange mutation. We firstly select two random bits l,k $(l>k)$, and examine if $l-k=1$ and l is an even number. If so, we then select two random bits again until l,k don't satisfied these conditions.

F. Population Size and Generation

A large increasing population size can reduce the probability of getting to local optima for a GA, and the computation optimality will be increased greatly. So population size is taken as 30 in our experiments. In the bottom-up progress, since both the number of nodes in every level and the chromosome length of an individual tend to decrease, the maximum evolution generation for the GA is reduced accordingly.

3.4 Merging Scheme

In the pathlength delay model, the conventional bottom-up merging schemes choose the sub-tree pair with the minimum distance between their roots to merge, and the merging point is the center of the two roots. This method can only ensure local zero-skew because the delays from clock source to all sinks are not identical. We propose a novel style of merging scheme that can realize the global zero skew. Two adjacent nodes are lined in sequence according to a chromosome, and instead of choosing the center of the two nodes as the merging point, we compute the coordinates of merging point such that delays from each merging point to its all sinks are identical, as shown in Fig 5.

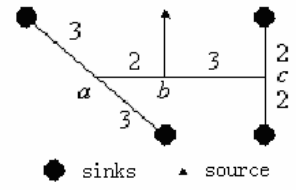


Fig 5. Merging scheme

There are four sinks and one source in Figure 5, internal nodes a,c are the centers of the lines between two sinks and the optimal result of the first level. The internal node b in the second level is not the center of the nodes a,c, but linear delays from b to all sinks are identical.

3.5 Odd Node Merging

When there is an odd node, the merged scheme should be modified in forming a binary tree. For example, there are 7 nodes in a chromosome as shown in Figure 6, when the first 6 nodes are merged, the last one is left alone. We propose to take an odd node directly as the next level's input node.

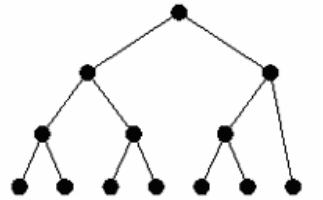


Fig 6. Odd node merging

3.6 Algorithm Stop Criteria

In all levels of the MLGA, the stop criteria are decided by the maximum evolution generations. Repeatedly execute genetic algorithms until the root of a binary tree, then, the total algorithm is terminated.

4 Experimental Results

Test problems are divided into two parts. The first part consists of some random cases in which some clock sinks and one clock source are created randomly in a given area. For example, c64 denotes that the clock network has 64 clock sinks and one clock source with their corresponding coordinates. The second part includes some standard benchmark data files r1~r5, which include clock sinks, clock source and their coordinate information (see Table 1).

Table 1: Parameters for test cases r1~r5

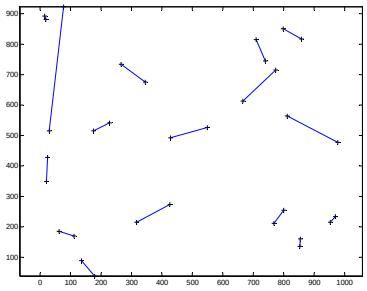
File name	*r1*	*r2*	*r3*	*r4*	*r5*
Sink number	267	598	862	1903	3101

The MLGA proposed in this paper are implemented using Matlab6.0 language on a PC (1 CPU, Intel P4 2.0GHz, 512MB RAM, 60GB Hard Disk) with the windows XP operating system. First we implement our algorithm on random case c32 so as to illustrate the constructing procedure of the clock binary tree. In the experimental implementation, the population size is 30, the crossover probability is 0.6 and the mutation probability is 0.02 in the MLGA. The total procedure is divided into 5 levels as shown in Fig 7, and maximum generations for all level are shown in Table 2

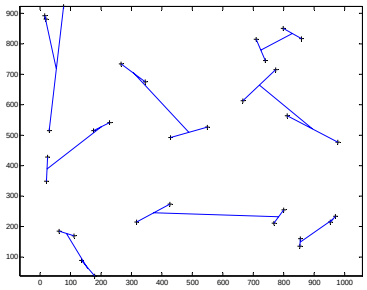
Table 2: Levels and maximum generations for c32

Level number	*1*	*2*	*3*	*4*	*5*
Generation	1000	500	250	125	63

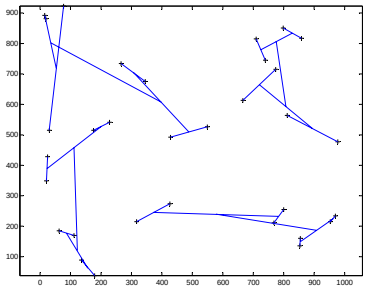
The Fig. 7 shows that the MLGA is able to construct a clock binary tree. In order to analyze the advantage of this method in total pathlength and running time, we compare the results got by the MLGA with that of the bottom-up matching approach. We generate randomly 6 test cases such as c16, c32, c64, c100, c128, and c150. For all the cases, the population size is 30, the crossover probability is 0.6, and the mutation probability is 0.02 in MLGA.



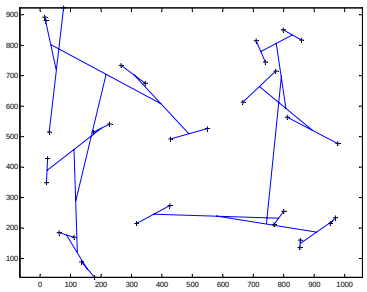
(a) Level one



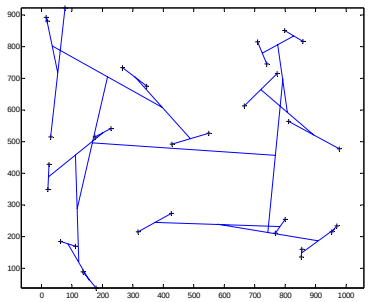
(b) Level two



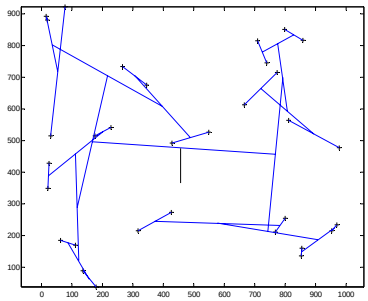
(c) Level three



(d) Level four



(e) Level five



(f) Final level

Fig 7. Constructing procedure of a clock binary tree base on the MLGA

The maximum evolutional generations of the MLGA in all levels can be designed as follows. Since the chromosome length decreases from one level to the next level, the number of generation should be decreased. The number of generation in $(i+1)th$ level is only half of that in the i th level. For test case c32, generations in these levels are 1000, 500, 250, 125, 63. Table 3,4 lists the results by the two algorithms, and the result in each case is the best one in 10 independent runs.

Table 3: Results By MLGA

Test Case	*MLGA*		
	Path-length (um)	*Time (s)*	*Zero skew*
c16	36.8	23.2	y
c32	52.9	41.2	y
c64	153.4	77.3	y
c100	278.6	130.1	y
c128	291.7	137.2	y
c150	323.9	162.5	y

Table 4: Results by the bottom-up matching approach

Test Case	*Bottom-up Matching Approach*		
	Path-length (um)	*Time (s)*	*Zero skew*
c16	37.5	0.03	n
c32	54.4	0.047	n
c64	163.6	0.062	n
c100	298.5	0.127	n
c128	313.4	0.143	n
c150	348.6	0.171	n

Table 3 and Table 4 show that pathlength got by the MLGA is superior to that of the bottom-up matching approach, but the time consumed is much longer. In order to decrease running time of the MLGA, we modify the decreasing strategies of every level's maximum generations as 1/2, 1/3, 1/4, where 1/j decreasing strategies means that the number of generation in $(i+1)th$ level is 1/j of that the i th level. Take c32 as an example, the first level's evolutional generation, the decreasing strategy, pathlength, and time consumed are shown in Table 5.

Table 5: Results of different maximum generations and decreasing strategies

The first level's evolutional generation	*1000*			*1500*		
Decreasing strategy	1/2	1/3	1/4	1/2	1/3	1/4
Pathlength (um)	52.7	52.8	52.9	51.3	51.7	51.7
Running time (s)	41.2	34.3	21.8	60.6	46.1	25.6

It can be seen from the Table 5 that the total pathlength has no obvious change with different decreasing strategies at a given maximum generation, but running time is reduced greatly with 1/2, 1/3, 1/4 decreasing strategy. Since the chromosome length is shortened at each level, the searching space is shrunk accordingly. By referring to the running time, the 1/4 decreasing strategy is preferred to other two.

The MLGA with 1/4 decreasing strategy is used to solve the standard benchmark files r1~r5. The pathlength results are listed in Table 6, and are compared with the pathlength results achieved by MMM, KCR, and the improved DME algorithm [2]. We don't compare the running time because of different computing platforms used in experiments.

Table 6: Comparison of pathlength by different methods

Bench-mark	*MMM method*	*KCR method*	*Improved DME*	*MLGA*
r1	1815	1627	1497	1511
r2	3625	3349	3013	3001
r3	4643	4360	3902	3877
r4	9376	8580	7782	7773
r5	13805	12928	11665	11632

Table 6 shows clearly that the total pathlength achieved by the MLGA with 1/4 decreasing strategy is much better than that of the conventional heuristic algorithms in linear pathlength model for most cases (especially for large cases) but r1. Since r1 has comparatively less nodes than other cases, the improved DME algorithm can also achieve better pathlength.

5 Conclusions

A multi-level genetic algorithm for construction of the clock binary tree is presented in this paper, which may yield less pathlength results than that of the existing algorithms while the zero skew is achieved with a heuristic merging scheme. In the MLGA, the encoding, fitness, genetic operators, and multi-level model are addressed in detail. Other key techniques, such as merging scheme, odd nodes proposal, generation decreasing strategy, are also designed properly. From the experiments on random test cases and standard benchmark test cases, it can be concluded that the MLGA is able to produce much better results compared with conventional heuristic algorithms.

Acknowledgments

This research was supported by the National Science Foundation of China (No.70571057) and Postdoctoral Science fund of China (No. 2005038151).

References

[1] Naveed A.Sherwani, Algorithms for VLSI Physical Design Automation, Kluwer Academic Publishers, ISBN 0-7923-9294-9

[2] Ting-Hai Chao, Yu-Chin Hsu, Jan-Ming Ho, Kahng A B, Zero skew clock routing with minimum wirelength, Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on, 1992, 39(11): 799-814

[3] Cong J, Kahng A B, Koh C K, Tsao C W, Bounded-skew clock and Steiner routing under Elmore delay, Computer-Aided Design, 1995. ICCAD-95.1995: 66-71

[4] Jackson M A B, Srinivasan A, Kuh E S, Clock routing for high-performance Ics, Design Automation Conference, 1990: 573~579

[5] Kahng A B, Cong J, Robins G, high-performance clock routing based on recursive geometric matching, in proc ACM/IEEE Design Automation conf, 1991: 322-327

[6] Edahiro M, Delay Minimization For Zero-skew Routing, Computer-Aided Design, 1993. ICCAD-93. Digest of Technical Papers, 1993 IEEE/ACM International Conference on, 1993: 563-566

[7] Chung-Wen Albert Tsao and Cheng-Kok Koh, UST/DME: A Clock Tree Router for General Skew Constraints,' ACM Trans.on Design Automation of Electronic Systems, 2002, 7(3): 359-379

[8] Kahng A B, Tsao C W A, Planar-DME: Improved planar zeroskew clock routing with minimum pathlength delay, in Proc. European Design Automation Conf, 1994: 440-445

[9] Balakrishnan P V, Gupta R, Jacob V S, Development of hybrid genetic algorithms for product line designs, Systems, Man and Cybernetics, Part B, IEEE Transactions on, 2004, 34(1): 468-483

[10] Potts J C, Giddens T D, Yadav S B, The development and evaluation of an improved genetic algorithm based on migration and artificial selection, Systems, Man and Cybernetics, IEEE Transactions on, 1994, 24 (1): 73-86

[11] Ji Hun Koo, Tae Seon Kim, Sung Soo Dong, Chong Ho Lee, Development of FPGA based adaptive image enhancement filter system using genetic algorithms, Evolutionary, CEC '02. Proceedings of the 2002 Congress on, 2002, 2:1480-1485

[12] S. Yang. Adaptive mutation using statistics mechanism for genetic algorithms. In F. Coenen, A. Preece and A. Macintosh (editors), Research and Development in Intelligent Systems, pp. 19-32, 2003. London: Springer-Verlag

[13] S. Yang. Memory-based immigrants for genetic algorithms in dynamic environments. Proceedings of the 2005 Genetic and Evolutionary Computation Conference, Vol. 2, pp. 1115-1122, 2005. ACM Press.

[14] Moon BR, Lee YS, Kim CK, GEORG : VLSI Circuit Partitioner with a New Genetic Algorithm Framework. Journal of Intelligent Manufacturing, Vol. 9, pp.401-12, 1998

[15] S. Areibi, An Integrated Genetic Algorithm With Dynamic Hill Climbining for VLSI Circuit Partitioning, IEEE Genetic and Evolutionary Computation Conference, pp: 97-102, July 2000

[16] Y. S. Ong, P.B. Nair and A.J. Keane, "Evolutionary Optimization of Computationally Expensive Problems via Surrogate Modeling", American Institute of Aeronautics and Astronautics Journal, 2003, Vol. 41, No. 4, pp. 687-696

[17] Yaochu Jin, Bernhard Sendhoff, Edgar Körner: Evolutionary Multi-objective Optimization for Simultaneous Generation of Signal-Type and Symbol-Type Representations. EMO 2005: 752-766

[18] S. Yang. Constructing dynamic test environments for genetic algorithms based on problem difficulty. Proceedings of the 2004 Congress on Evolutionary Computation, Vol. 2, pp. 1262-1269, 2004

Nan Guofang, received the B.S degrees in Department of Autamation from Beijing Institute of Technology in 1998, and M.S. degree in School of Automation from Tianjin University in 2002, he received his Ph.D in the Institute of Systems Engineering from Tianjin University in 2004. His now research interests include evolutionary computation, VLSI CAD, Information retrieval, evolvable hardware and related topics.