Evolving Diverse Hardwares Using Speciated Genetic Algorithm

Keum-Sung Hwang
Dept. of Computer Science, Yonsei University
134 Shinchon-dong, Sudaemoon-ku,
Seoul 120-749, Korea
yellowg@cs.yonsei.ac.kr

Sung-Bae Cho
Dept. of Computer Science, Yonsei University
134 Shinchon-dong, Sudaemoon-ku,
Seoul 120-749, Korea
sbcho@cs.yonsei.ac.kr

Abstract - Evolvable Hardware (EHW) is an attractive topic recently because it can reconfigure itself to adapt to the environment embedded. An EHW uses genetic algorithm, one of evolutionary algorithms to search the goal hardware. In this paper, we propose an EHW using speciated GA that can evolve diverse circuits with one-step evolution. Speciation algorithm helps finding diverse solutions as the result of the evolution and keeps the diversity during the evolution. We have applied fitness sharing method for speciation, to the EHW of 6-multiplexer, and have got diverse hardware structures. Also, we have found the circuit with 35% earlier generation than conventional genetic algorithm.

I. INTRODUCTION

Evolvable hardware (EHW), which is studied actively in recent years, has got increasing attention since early 1990's with the development of easily reconfigurable hardware such as field programmable gate arrays (FPGAs). EHW has many advantages such as adaptation to environment and fault tolerance, so many researchers are working on EHW [1].

EHW can adapt itself to unknown environment using architecture features of FPGA that enables hardware to be reconfigured [2][3]. It uses genetic learning method based on evolutionary algorithm (EA) to search goal hardware function [3][4]. It converts the architecture data of hardware to chromosomes, evolves the hardware structures till the goal function is found. The chromosomes' fitness values are gotten from the similarity between the function and goal function.

In EHW, Genetic Algorithm (GA) determines the capacity. The GA's searching capacity of conventional EHW is not so good, because it does not maintain the diversity of population well, while the number of hardware structure with same function is large. EHW has many GA-deceptions on fitness landscape. It does not guarantee similar performance as much as the similarity of hardware structure though we may find a structure similar to the goal structure. This causes long convergence time and escape time from genetic drift.

Niching methods have been developed to reduce the effect of genetic drift. They maintain population diversity and broaden the GA's parallel search space. They also prevent the GA from being trapped in local optima of the search space. Niching methods are based on the mechanics of natural ecosystems [5]. In nature, there are many different species defined as a group of individuals with similar biological features capable of interbreeding among themselves. For each niche, the resources are shared among the population of that niche. Niching GA causes a natural emergence and leads to survived niches and species. A niche is commonly referred to as an optimum of the domain, the fitness representing the resources of that niche. Species can be defined as similar individuals in terms of similarity metrics. Since this mechanism, niching is also called as speciation.

In this paper, we have applied the speciation method to GA of EHW. The method leads to good performance of EHW, i.e., getting diverse characteristic hardwares, fast convergence and non-distributed (even) searching-capacity. Especially, creativity of speciated EHW is very interesting. The hardware structures from evolution have diverse structural characteristics. If they are fully analyzed, utilized and combined, we can get more powerful hardware.

Section 2 presents a model of EHW and fitness sharing method. Section 3 is devoted to speciated EHW schemes proposed. Section 4 investigates the evolution of speciated EHW on test problems defined in Section 3 and compares their efficiency with the conventional EHW using standard GA.

II. BACKGROUNDS

A. Evolvable Hardware Device

We discuss the genetic learning component simulated in our experiments. We select a test device based on that used by T. Higuchi in [2]. It is GAL16V8 chip (Fig. 1). The GAL16V8 chip is an example of Field Programmable Gate Array (FPGA). It consists of 8 logic cells, which are called OLMCs (Output Logic Macro Cells), and their interconnections consisted of fuse array. A logic cell performs some logical functions. Its logic function can be selected by specifying special bits in the device. The fuse array is used to determine the interconnections between device inputs and logic cells as well as to specify the logic cells' AND-term inputs. If a link on a particular row of a fuse array is set to "connected," then the corresponding input signal is connected to the row. The logic cell has the

0-7803-7282-4/02/$10.00 ©2002 IEEE

structure as shown in Fig. 2. It contains 5 possible operational modes such as "registered device" or "combinatorial output device" by specifying 3 special bits (SYN, AC0, AC1). Thus, an arbitrary Boolean function can be configured by determining both the links of the fuse array pattern and the function of the logic cell.

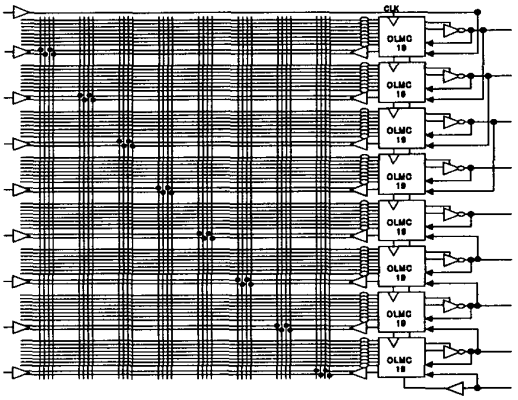

Fig. 1. Logic diagram of the GAL16V8 chip

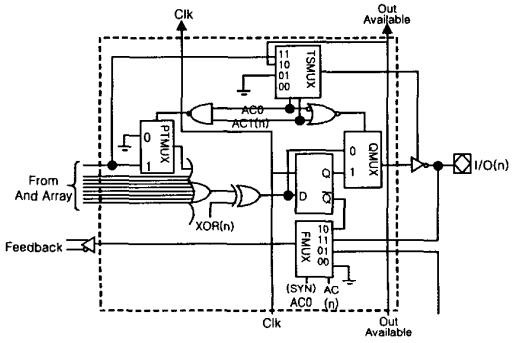

Fig. 2. Output Logic Macro Cell (OLMC) structure

A standard concept of EHW is to find the architecture bits (fuse bits array and OLMC specifying bits) of goal hardware, which operates well on a required condition, by using GA learning [3].

B. Speciation Method

Speciation method is the technique to form and maintain multiple species in a population [6]. Some speciation methods restrict mating of populations to that with similar individuals, and other methods manipulate the fitness values of species to control selection pressure. Especially, the latter methods are performed by niching method naturally. In this paper, we have used one of niching mehods, fitness sharing.

Fitness sharing, as introduced by Goldberg and Richardson (1987), is a fitness scaling mechanism, which alters only the fitness evaluation stage of a GA. The idea behind fitness sharing is as follows. If similar individuals (species) share fitness (resources), then the number of individuals that can reside in any one region of the fitness landscape is limited. In the result, diverse speciation can survive fairly by sharing [5].

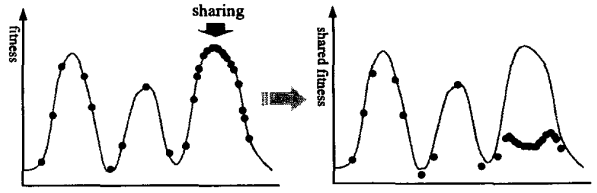

Fig. 3. Sharing in fitness landscape

Fitness sharing modifies the search landscape by reducing the fitness in densely populated regions. It lowers each population member's fitness by an amount, which is nearly equal to the number of similar individuals in the population. Typically, the shared fitness sf_i of an individual i with fitness f_i is simply

$$sf_i = \frac{f_i}{m_i}$$

where m_i is the niche count, which measures the approximate number of individuals, with which the fitness f_i is shared. The niche count is calculated by summing a sharing function value over all members of the population

$$m_i = \sum_{j=1}^{N} sh(d_{ij})$$

where N denotes the population size and d_{ij} represents the distance between the individuals i and j. Thence, the sharing function (sh) measures the similarity between two population members. It returns a '1' if the members are identical, a '0' if they cross some threshold of dissimilarity, and an intermediate value for intermediate levels of dissimilarity. A common sharing function is given as follows:

$$sh(d_{ij}) = \begin{cases} 1-(\frac{d_{ij}}{\sigma_s})^{\alpha}, & for\ 0 \le d_{ij} < \sigma_s \\ 0, & for\ d_{ij} \ge \sigma_s \end{cases}$$

where σ_s denotes the threshold of dissimilarity (sharing radius) and α is a constant that regulates the shape of the sharing function. α is commonly set to one with the resulting sharing function referred to as the triangular sharing function [4].

The distance d_{ij} between two individuals i and j is characterized by a similarity metric based on either genotypic or phenotypic similarity. Genotypic similarity is related to bit-string representation. It is generally Hamming distance, which is the number of bits that do not match when comparing two strings. Phenotypic similarity is directly linked to real parameters of the search space. It can be the Euclidian distance for instance. Sharing based on phenotypic similarity may give slightly better results than sharing with genotypic similarity [7].

III. SPECIATED EVOLVABLE HARDWARE

Speciated EHW has one additional process on EHW. The process is as shown in Fig. 4. Fitness sharing process is appended before genetic operation (especially, selection operator) and after the fitness evaluation, since the evaluation uses raw fitness value.

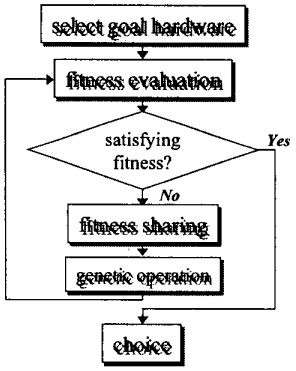

Fig. 4. Process of speciated EHW

C. Hardware Simulation

We have simulated the simplified GAL chip that have only one OLMC (Fig. 5). The OLMC's function is fixed "combinatorial output device" for simple simulation. This hardware allows at most 8 OR-combination, which is composed by at most 6 AND-combination.

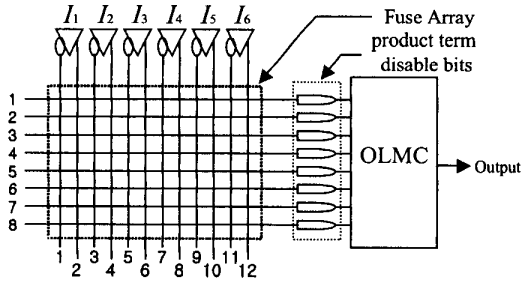

Fig. 5. A simplified GAL structure

D. Chromosome Encoding

In Fig.5, numbers 1~8 are inputs of OLMC and I_1~I_6 are inputs of the device. There are fuse array that contain 96 (8×6×2) bits that specify the OLMC's AND-term inputs, and 8 product-term-disable bits that determine the usability of OLMC's input. The fuse bits and product-term-disable bits compose a chromosome, structure of which is shown in Fig. 6.

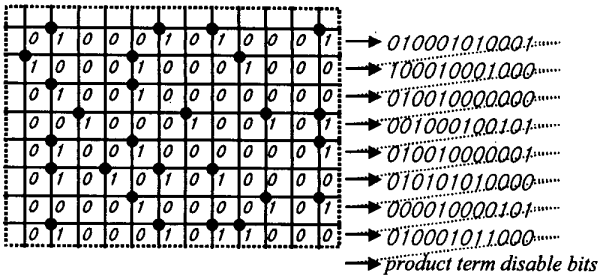

Fig. 6. Example of fuse array

In the fuse array, a couple of bits '11' of any one input are useless, because it means

$$I_k \text{ AND (NOT } I_k) = 0, \qquad \text{where } k=1,2,\ldots,6$$

Hence the couple of bits can be converted to 3 character strings like Table 1. This reduces chromosome's length (finally, 8×6+8=56 characters), and helps fast convergence.

TABLE 1. SIMPLIFICATION OF FUSE BITS

Couple Of Bits	Replaced Character
00	'0'
01	'1'
10	'2'
11	not available

E. Fitness Evaluation

We have selected the goal hardware as 6-multiplexer, which is simple hardware of 4 input bits, 2 selection bits and one output bit. It is good test hardware because it can be made by using only AND gates and OR gates, and it can be understood and analyzed easily.

To evaluate the fitness of evolved hardware, we have compared the patterns of inputs and outputs of the hardware and the goal function. The process of evaluation is shown in Fig. 7.

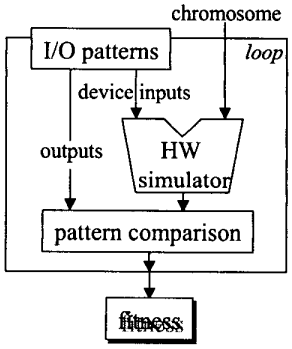

Fig. 7. Process of fitness evaluation

The fitness value of hardware is the probability of matching patterns' number as follows. In this case, maximum fitness value is 100.

$$fitness = \frac{\text{number of matched patterns}}{\text{number of all patterns}} \times 100$$

F. Shared Fitness Evaluation

The shared fitness of individual i (sf_i) is computed as follows. This is simplified formula of the original one.

$$sf_i = \frac{f_i}{\sum_{j=1}^{N} \begin{cases} 1 - \frac{d_{ij}}{\sigma_s}, & \text{for } 0 \le d_{ij} < \sigma_s \\ 0, & \text{for } d_{ij} \ge \sigma_s \end{cases}}$$

The distance between individuals is calculated using both types, i.e., genotypic distance and phenotypic distance. The distance is as follows.

$$d_{ij} = \sqrt[2]{\sum_{k=1}^{8} (h_k(i,j))^2}$$

In our experiments, a chromosome has 8 blocks, each of which represents product-term of OLMC input. Hence each block has distinct characteristic. We have used Hamming distance $h_k(i,j)$ that means distance between k'th blocks of individuals i and j. Then, the distances $h_k(i,j)$ have been combined to d_{ij} by the similarity metric based on phenotypic similarity, Euclidian distance.

A product-term-disable bit decides whether to use one OLMC input. Its difference can be considered as difference of the block of the product term. Hence, we give a maximum value '6' to the Hamming distance h_k when the k'th product-term-disable bit is different between individuals i and j.

G. Genetic Operators

We have used a selection method different from roulette wheel selection. The selection is challenged to all individuals of the population at least one time, then the selection ratio (p_s) is considered to determine whether to take it or not. With this selection method, the individual of the best shared fitness is always selected and any other individuals have least one challenge of selection. The selection ratio is as follows.

$$p_s(i) = \frac{sf_i}{\max_{j=1,2,\cdots,N} sf_j}$$

We have used three crossover methods equally in the experiment. They are one point crossover, uniform crossover and block-unit uniform crossover. One point crossover is to perform crossover at a random point. Uniform crossover is to randomly confirm whether performing crossover between each bit of two chromosomes. It helps fast gene mixing.

Block-unit uniform crossover is the modified uniform crossover, which swaps blocks instead of bits. A chromosome has 8 product terms and 8 product-term-disable bits. They have independent information. The block-unit uniform crossover concerns the information. An example of the crossover is shown in Fig. 8.

parent 1	1200101100012210000001000210100102000211002012 10 | 11110100
parent 2	1210112001011000122100110010002101001020002110 02 | 00100011
mask	1.....0.....1.....1.....0.....0.....0.....1..... | 10110001
offspring 1	1210111100011000122100110210100102000211002110 02 | 01100101
offspring 2	1200102001012210000001000010002101001020002012 10 | 10110010

Fig. 8. Uniform crossover in the unit of product-term block

We have performed two mutation methods equally in experiments. The first method is to flip a product-term-disable bit without changing the product-term bits. The second method is to flip a product-term-disable bit with reseting the product-tem bits.

IV. EXPERIMENTAL RESULTS

All experiments have been performed with the experimental parameters in Table 2. The values have been chosen empirically and elitism have not been used. Among the parameters, solution-search rate is a stop condition for the speciation method. The evolution is stopped when the ratio of solutions (circuits) to population reaches to the solution-search rate. In this experiments, the number of solutions is 15 by the multiplication of the population size 50 and the solution-search rate 0.3.

TABLE 2. PARAMETERS OF EXPERIMENTS

Parameters	Values
population size	50
mutation rate	0.02
crossover rate	0.4
sharing radius	3.0
solution-search rate	0.3

We have investigated the capacity of searching multiple solutions. Fig. 9 shows general change of the number of solutions with the exeption of redundant chromosomes during the generation. The numbers are increased steadily after one solution is found.

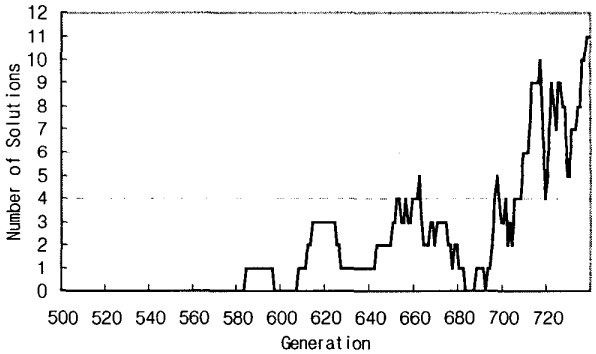

Fig. 9. The number of solutions are increasing during the generations

We have observed the degree of fitness sharing (Fig. 10). The average fitness values are going under the raw fitness values. This means fitness sharing and speciation are operating.

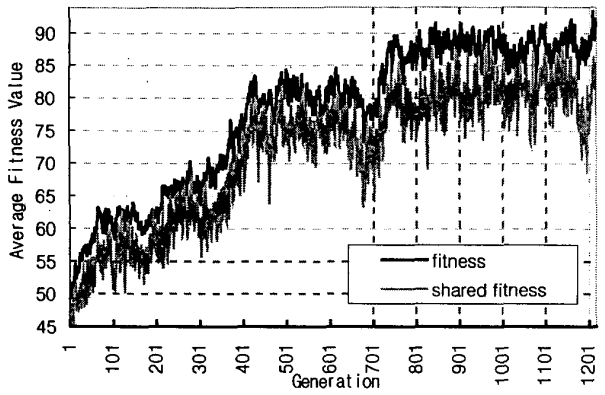

Fig. 10. Average fitness values of evolution of 6 multiplexer

We have investigated the maximum fitness values during the generations in Fig. 11. The first solution appears at the 901st generation. The number of solutions is satisfied with the solution-search rate at the 1,216th generation.

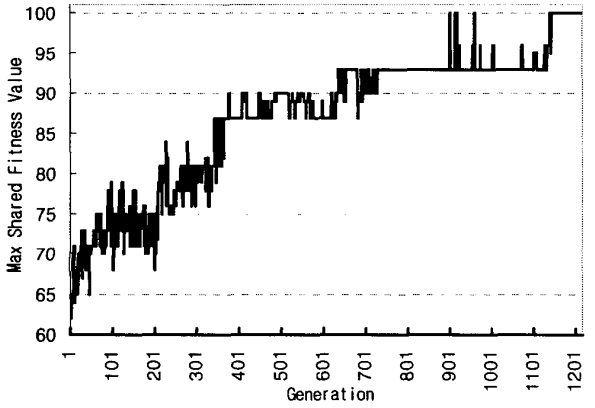

Fig. 11. Maximum fitness transition with sharing

In the result of the experiment, 13 chromosomes with the except of 2 redundant ones are obtained. (The number of solution-search is 15). All of these 13 chromosomes have different structures, but have the same hardware function, i.e., 6 multiplexer. To analyze the difference of solutions, we have conducted single linkage clustering and depicted the dendrogram [8]. There are two dendrograms in Fig. 12. One is gotten from the solutions evolved by conventional GA, and the other is gotten from the solutions evolved by speciated GA. The dissimilarity of speciated GA is more distinct than that of conventional GA.

The distance values between the solutions are distributed from 3.3 to 6.7. The distance is 2.4 when a block is completely different, and the distance is 8.5 when two blocks are completely different. From this, we can observe that the solutions have at least one-block-complete-difference and at most two-block-complete-difference.

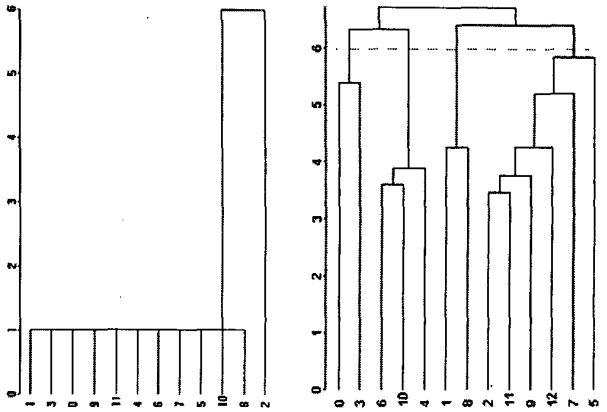

Fig. 12. Dendrograms of evolved circuits with single linkage clustering

We have divided the solutions into 4 groups at the distance point 6. The cut-line is depicted by dotted line in Fig. 12. Then, we have drawed 4 circuits that represent the groups (Fig.13).

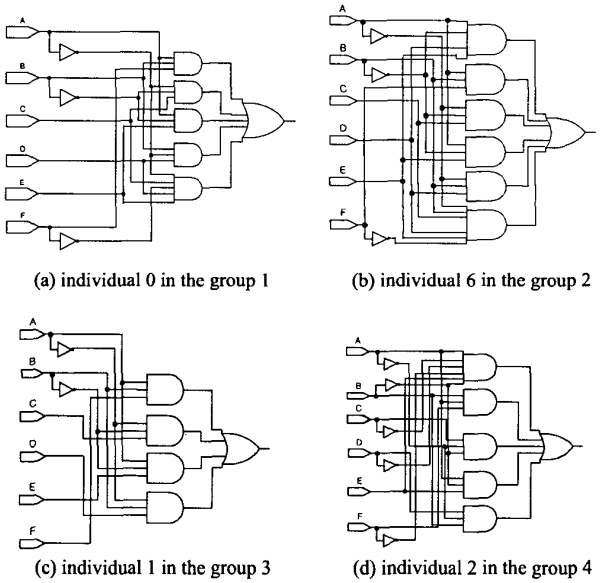

Fig. 13. The circuit structures of representatives of species

Each circuit has not only characteristic structure but also same substructures that operate as same function. For instance, AND gates from the 2nd to the 5th of individual 6 are same with all AND gates of individual 1. In Fig. 14, every circuit has the same substructure.

Hence, the gates with the except of the same structure are extra functional parts of the circuit, e.g., individual 0 has extra function of $\overline{A}CDE\overline{F}$. We may find useful extra functions of them.

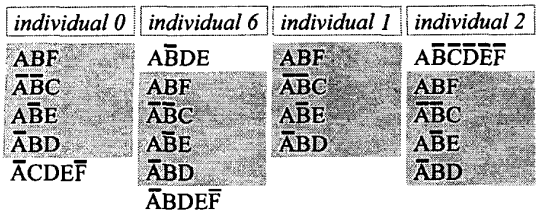

Fig. 14. The AND-terms of circuits
(grey boxes denote the same substructure)

In this experiment, the extra parts are trivial. However, they may be able to have extra strong functions in a large hardware device. This research aims to get the creative and novel hardware designs.

Getting diverse circuits has another benefit of that we can find the better hardware in any criteria. e.g., the individual 1 uses the fewest gates and input signals as shown in Table 3. It means that the circuit 1 has the simplest structure, and we can select it as optimized one. In other case, if you want a circuit that overcome hard benchmarks, you can test the circuits without evaluation during the evolution. It will saves much time.

TABLE 3. COMPARISON OF CIRCUIT STRUCTURES. (IN EACH COMPARISON, THE SMALLER NUMBER IS THE BETTER.)

	Individual 0	Individual 6	Individual 1	Individual 2
# of gate inputs	17	17	12	18
# of OLMC inputs	5	6	4	5
# of used inputs	9	9	8	11

Diverse species can serve one more benefit. When the first entrance of a logic cell is out of order, the circuit 1 cannot work well. But we can replace it with circuit 2 that works well without first AND-term, if we have the circuit species. Diverse circuits can be diverse solutions of a problem.

On the other hand, we have compared the speed of finding solutions of speciated GA with conventional GA. Because speciation algorithm keeps the divisity well, it is favorable to searching solutions. We have conducted 50 runs, which are performed till finding the first solution. The result is as shown in Table 4.

TABLE 4. THE COMPARISON OF GENERATIONS TO FIND THE FIRST SOLUTION OF CONVENTIONAL GA AND SPECIATED GA

	Conventional GA	Speciated GA
Average generation	1147.16	747.46
Fastest generation	143	77
Slowest generation	5373	2485
Standard deviation	1061.35	584.70

The searching capacity of speciated GA is better as expected. The generation of obtaining the first solution of speciation algorithm is earlier and its standard distribution is smaller. This means that the algorithm has an even good searching capacity.

V. CONCLUDING REMARKS

In this paper, we have confirmed many advantages of EHW using speciatied GA. With obtaining diverse hardwares, we can see the availability of superior extra hardwares as well as preparations for fault situation. We can also see that the searching capacity of spiceated EHW is superior to conventional one. The EHW research using speciation technique have interested us.

However, the running speed of fitness sharing method is not so good because of its complexity. We will solve the problem by using other speciation methods or other techniques. Additionally, we will perform more experiments in large scale EHW and search the methods to utilize the species efficiently, e.g. combination. They can contribute a novel evolutionary hardware design technique.

References

[1] X. Yao and T. Higuchi, "Promises and challenges of evolvable hardware," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 29, pp. 87-97, February 1999.

[2] T. Higuchi, H. Iba, and B. Manderick, "Evolvable hardware," *Massively Parallel Artificial Intelligence*, pp. 398-421, MIT Press, 1994.

[3] T. Higuchi, T. Niwa, T. Tanaka, H. Iba, H. de Garis, and T. Furuya, "Evolving hardware with genetic learning," *Proc. of Simulation of Adaptive Behavior*, pp. 417-424, MIT Press, 1992.

[4] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, 1989.

[5] B. Sareni, L. Krahenbuhl, "Fitness sharing and niching methods revisited," *IEEE Trans. on Evolutionary Computation*, vol. 2, no. 3, pp. 97-106, September 1998.

[6] K. Deb and W. M. Spears, "Speciation methods," *Evolutionary Computation 2, Advanced Algorithms and Operators*, Institute of Physics Publishing, Ch. 14, 2000.

[7] K. Deb and D. E. Goldberg, "An investigation of niche and species formation in genetic function optimization," *Proc. 3rd Int. Conf. Genetic Algorithms*, J. D. Schaffer, ed. San Mateo, CA: Morgan Kaufmann, pp. 42-50, 1989.

[8] S. W. Mahfoud, "Niching methods," *Evolutionary Computation 2, Advanced Algorithms and Operators*, Institute of Physics Publishing, pp. 87-92, 2000.

[9] P. Darwen and X. Yao, "Every niching method has its niche: Fitness sharing and implicit sharing compared," *Proc. of Parallel Problem Solving from Nature (PPSN) IV*, vol.1141, Lecture Notes in Computer Science, Springer-Verlag, Berlin, pp. 398-407, 1996.

[10] J. R. Koza, "Evolution of subsumption using genetic programming," *Toward a Practice of Autonomous Systems*, pp. 110-119, MIT Press, 1993.

[11] D. Goldberg, K. Deb, and J. Horn, "Massive multimodality, deception, and genetic algorithms," *In Parallel Problem Solving from Nature 2*, pp. 37-46, North-Holland, 1992.

[12] R. E. Smith, S. Forrest, and A. S. Perelson, "Searching for diverse, cooperative populations with genetic algorithms," *Evolutionary Computation*, vol. 1, no. 2, pp. 127-149, 1993.