Improving evolvable hardware by applying the speciation technique

Keum-Sung Hwang, Sung-Bae Cho*

Department of Computer Science, Yonsei University, 262 Seongsan-ro, Seodaemun-gu, Seoul 120-749, Republic of Korea

ARTICLE INFO

Article history:
Received 29 November 2005
Received in revised form 19 October 2007
Accepted 9 January 2008
Available online 22 April 2008

Keywords:
Evolvable hardware
Speciation technique
Fitness sharing
Extra function
Digital circuit design

ABSTRACT

Evolvable hardware (EHW) has recently become a highly attractive topic of study because it offers a way of adapting hardware to a given embedded environment. However, it is not easy to evolve hardware efficiently and effectively, so many challenges continue to exist when trying to solve problems. In this paper, we propose a method that uses the speciation technique to enable diverse circuits to evolve efficiently by the process of one-step evolution. As a result of studying the landscape contained in the EHW example, we have found complicated spaces contain many peaks that can lead to deceptions when using the evolving process, and the speciation technique profits from the evolution of EHW. We also studied that the speciated hardware ensemble might be a good candidate for more complex and rigorous function. In the experiments, we applied the fitness sharing method as the speciation technique, and obtained diverse hardware modules, then ascertained the efficiency of these structures. We also show that several useful extra functions and better overall performance can be obtained by analyzing diverse circuits with the speciation technique.

© 2008 Elsevier B.V. All rights reserved.

1. Introduction

Evolvable hardware (EHW), which has been studied actively in recent years, has received increasing attention since the early 1990s with the development of easily reconfigurable hardware known as field programmable gate arrays (FPGAs). EHW offers many advantages, including adaptation to environment and fault tolerance. This has led to studies by many researchers in the field of EHW [1–4].

EHW can adapt itself to unknown environments using the architectural features of FPGAs. These features enable hardware to be reconfigured. This process uses a genetic learning method based on an evolutionary algorithm (EA) to search for the goal hardware function [5–9]. It converts the architectural data of the hardware into chromosomes and evolves the hardware structures until the goal function is found. The chromosomes' fitness values are then obtained from the similarity between the function and goal function.

The evolutionary algorithm used in EHW usually determines performance; an ordinary EA is the genetic algorithm (GA). However, the searching capacity of the GA on EHW is not so good, because it does not maintain the diversity of population well. Also, the number of hardware structures with the same function is large, so there are many GA-deceptions within the fitness landscape. Finally, this method does not guarantee similarity of performance as much as similarity of the hardware structures, although we may find a hardware structure similar to the goal structure. This factor causes long periods of convergence time as well as escape time from genetic drift.

Niching methods have been developed in order to reduce the losses that result from genetic drift. These methods maintain the population diversity and broaden the GAs parallel search space. They also prevent the GA from being trapped in the local optima of the search space. Niching methods are based on the mechanics of natural ecosystems [10]. In nature, there exist many different species; each defined as a group of individuals with similar biological features capable of interbreeding among themselves. For each niche, resources are shared among the population of that niche. Niching of GAs causes a natural emergence and leads to survived niches and species. A niche is commonly referred to as an optimum of the domain, with fitness representing the resources of that niche. Species can be defined as similar individuals in terms of similarity metrics. Using this mechanism, most niching methods have also come to be known as speciation techniques in recent years.

In this paper, we study the features of hardware evolution, especially the fitness landscape. We utilized a speciation technique. It is proposed that the speciation method, when applied to EHW, will enhance performance by obtaining diverse
characteristic hardware, fast evolution and an even (not rugged) searching capacity. Especially, the creativity of the speciated EHW is interesting; it is expected that the hardware structures that evolve from the speciated GA will have diverse structural characteristics. Furthermore, the speciated EHW can be the good solution to solve more complex and rigorous problems. For example, an EHW for spaceship has to pass many strict and complex tests, but it is mostly impossible to evolve with the tests because it costs too much time. The speciated EHW ensemble can be the well-prepared candidates for the tests. In this paper, we discuss about it.

The speciation technique has already been applied to digital circuit design by McKay [11] and Hwang and Cho [12]. The main contribution of them was to improve the evolutionary performance by using a speciation technique. In the research of McKay, the speciation algorithm was adopted on genetic programming (GP), which was not easy to embody on hardware chips. The paper is an expanded version of EHW research by Hwang and Cho [12], and we have focused on dealing with how to use the speciation technique and improve the performance by the ensemble of speciated hardware modules.

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

2. Backgrounds

2.1. The evolvable hardware device

In this section, we discuss the genetic learning component that we simulated in our experiments. We selected a test device based on that used by Higuchi et al. [13]. It is known as a GAL16V8 chip (Fig. 1). The GAL16V8 chip is an example of a Field Programmable Gate Array (FPGA). It consists of eight logic cells, called OLMCs (Output Logic Macro Cells), and their interconnections consist of fuse arrays. A logic cell is able to perform logical functions, which can be selected by specifying special bits to the device. Fuse arrays are used to determine the interconnections between the device inputs and the 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 structure of the logic cell can be seen in Fig. 2. It contains five possible operational modes, for example, “registered device” or “combinatorial output device”. These are obtained by specifying three special bits (SYN, A0C, A1C). 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. A standard concept used by EHW is finding the architectural bits (including the fuse bits arrays and OLMC specifying bits) of the goal hardware. This process operates well under certain required conditions, for example, when using GA learning [6].

2.2. The speciation method

The speciation method refers to the process of forming and maintaining multiple species in a population [14]. Some speciation methods restrict the mating of individuals in certain populations with similar individuals, and other methods manipulate the fitness values of the given species in order to control selection pressure. Especially, the latter methods use a process called niching. In this paper, we used one such niching method, known as fitness sharing.

Fitness sharing, as introduced by Goldberg and Richardson (1987) [15], is a fitness scaling mechanism, which alters only the fitness evaluation stage of a given 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. As a result, diverse speciation allows individuals to survive fairly by sharing [10].

Fitness sharing modifies the search landscape by reducing fitness in densely populated regions. It lowers each population member’s fitness by a given amount, which is nearly equal to the fitness in densely populated regions. It lowers each population member’s fitness by a given amount, which is nearly equal to the fitness of similar individuals in the entire population. Typically, the shared fitness \(s f_i\) of an individual \(i\) with fitness \(f_i\) is simply

\[
s f_i = \frac{f_i}{m_i},
\]

where \(m_i\) is the niche count, which measures the approximate number of individuals, with which fitness \(f_i\) is shared. The niche count is calculated by summing a sharing function value over all the 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 \(j\). Thus, the sharing function \(sh(\cdot)\) measures the level of similarity between two population members.

Fig. 1. A logic diagram of the GAL16V8 chip.
Fig. 2. The Output Logic Macro Cell (OLMC) structure.
It returns a '1' if the members are identical, a '0' if they cross some threshold of dissimilarity, and a value for intermediate levels of dissimilarity. A common sharing function is given as follows:

\[
sh(d_{ij}) = \begin{cases} 
1 - \left( \frac{d_{ij}}{\sigma_s} \right)^{\alpha}, & \text{for } 0 \leq d_{ij} < \sigma_s, \\
0, & \text{for } d_{ij} \geq \sigma_s,
\end{cases}
\]  

(3)

where \(\sigma_s\) denotes the threshold of dissimilarity (sharing radius) and \(\alpha\) is a constant that regulates the shape of the sharing function. \(\alpha\) is commonly set to one with the resulting sharing function referred to as the triangular sharing function [9].

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 known as the Hamming distance, which refers to the number of bits that do not match when comparing two strings. Phenotypic similarity is directly linked to real parameters of the search space. This can be the Euclidian distance, for instance. Sharing based on phenotypic similarity may produce slightly better results than sharing with genotypic similarity [16].

2.3. Fitness landscape analysis

In order to justify using the speciation method with the EHW, we studied the fitness landscapes of the EHW (Fig. 3). In the experiments, we used a very simplified GAL chip device as shown in Fig. 4. The device shown on the left side of Fig. 4 has two inputs, two OLMC entrances and no product-term-disable bits. The function of the OLMCs is fixed according to the OR composition. The right side of Fig. 4 shows the process of converting from a 2-bit string to a 3-character string of chromosomes in order to reduce the cost. This is explained in Section 3.2 in detail. We first simulated the hardware, and then analyzed the fitness landscape.

Fig. 5 shows a fitness landscape graph with a chromosome axis and we also set \(p(0) = 1\) for consistency. The left graph in Fig. 6 shows the ruggedness graph of the fitness landscape. In the figure,
the correlation Hamming distance of 2 apart is the smallest, and the value increases from 3 apart. This means that the solution in reference to the fitness similarity in the GA will not be smooth. We also observed the distribution of fitness values with the Hamming distance. This result is shown in Fig. 6, where one optimal chromosome ‘2112’ is used as the base of measurement. We can see that the distance to the other optimum is the longest; the 0 point is near 2, and the other points are rather far. This means that there are deceptions.

We divided a chromosome into two sub-genes (sub-gene 1: loci 1 and 2; sub-gene 2: loci 3 and 4) in order to observe them in detail because one chromosome has as many dimensions as the number of bits in its own phenotype, and the expressible dimension limit of the graph is 3. In Fig. 7, the fitness landscape is demonstrated by the graph with three dimensions and two axes of sub-genes. We can also see the peaks and deceptions in the figure. The plane figure is shown on the right side of Fig. 7. In the figure, the points around one optima point (6, S8) of the fitness landscape slope steeply, and the fitness values are 0.5 or below except for point (5, S7), which is 0.75. This means it is very difficult to search for the optimal points. The other optimum possesses the same conditions.

3. Speciated evolvable hardware

The speciated EHW contains one additional process compared to the EHW. This extra process is shown in Fig. 8. The fitness sharing process is appended before the genetic operation (especially, the selection operator) and also after the fitness evaluation, since the evaluation process uses a raw fitness value.

3.1. Hardware simulation

We simulated the simplified GAL chips with only one OLMC (Fig. 9). The OLMCs function is fixed and known as a “combinatorial output device” for simple simulation. This hardware allows at the most an 8-OR-combination, which is composed of by at the most 6-AND-combinations.
3.2. Chromosome encoding

In Fig. 10, numbers 1–8 refer to inputs of the OLMC and I₁–I₆ are inputs of the device. There are fuse arrays that contain 96 \((8 \times 6 \times 2)\) bits that specify the OLMCs AND-term inputs, and 8 product-term-disable bits that determine the usability of OLMCs inputs. The fuse bits and product-term-disable bits compose a chromosome, the structure of which is shown in Fig. 10.

In the fuse array, a couple of bits \('11'\) of any one input are useless, because it means \(I_k \text{AND} \neg I_k = 0\), \(k = 1, 2, \ldots, 6\). Hence these bits can be converted into three character strings as shown in Table 1. This reduces the chromosome length (eventually to \(8 \times 6 / 2 + 8 = 56\) characters), and leads to more rapid convergence.

3.3. Fitness evaluation

We selected the goal hardware as a 6-multiplexer, which is a simple form of hardware containing four input bits, two selection bits and one output bit. This is a good form of hardware for testing 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 the evolved hardware, we compared the patterns of inputs and outputs of the hardware and the goal function. The evaluation process is shown in Fig. 11.

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

\[
\text{fitness} = \frac{\text{number of matched patterns}}{\text{number of all patterns}} \times 100 \quad (9)
\]

3.4. Shared fitness evaluation

The shared fitness of individual \(i\) \((s_{fi})\) is computed as follows. This is a simplified formula of the original one.

\[
s_{fi} = \frac{\sum_{j=1}^{N} f_i}{\sum_{k=1}^{n} s_k}; \quad f_i = \sum_{j=1}^{N} \left\{ \begin{array}{ll} 1 - d_{ij}/\sigma_s, & \text{for } 0 \leq d_{ij} < \sigma_s \\ \ 0, & \text{for } d_{ij} \geq \sigma_s \end{array} \right. \quad (10)
\]

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

\[
d_{ij} = \sqrt{\sum_{k=1}^{8} (b_k(i,j))^2} \quad (11)
\]

In our experiments, the chromosome had eight blocks, each of which represented the product-term of the OLMC input. Hence each block had a distinct characteristic. We used the Hamming distance \(h_k(i,j)\) as the distance between the \(k\)th blocks of the individuals \(i\) and \(j\). Then, the distances \(h_k(i,j)\) were combined with \(d_{ij}\) by the similarity metric based on the phenotypic similarity known as the Euclidian distance.

We used a product-term-disable bit to decide whether to use one OLMC input. Its difference can be considered as the difference of the block of the product-term. We then gave a maximum value of ‘6’ to the Hamming distance \(h_k\) when the \(k\)th product-term-disable bit differed between the individuals \(i\) and \(j\).

3.5. Selection operator

The most general method among selection operators used on GA is known as a roulette wheel selection. This method performs the selection process by making a roulette wheel motion in proportion to the fitness of the space. The operation of selecting one point among \(n\) individuals, as in a roulette wheel, works \(n'\) times [3], where the selection rate of each individual is \(s_i\):

\[
s_i = \frac{s_{fi}}{\sum_{k=1}^{n} s_k}; \quad n' = n \times p_s, \quad (12)
\]

Table 1

<table>
<thead>
<tr>
<th>Couple of bits</th>
<th>Replaced character</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>‘0’</td>
</tr>
<tr>
<td>01</td>
<td>‘1’</td>
</tr>
<tr>
<td>10</td>
<td>‘2’</td>
</tr>
<tr>
<td>11</td>
<td>Not available</td>
</tr>
</tbody>
</table>
where $p_s$ is the value of the selection rate and $n_0$ is the number of individuals selected as the parents of the next generation. This selection method forms an important part of the GA process because it leads to the evolving of genes by the process of selecting superior individuals. However, this selection method may cause noise by selecting individuals that are almost similar or by selecting bad ones. Also, there is more chance of losing good genes and becoming trapped in the local optima. In order to prevent these problems, we used a proprietary distributed roulette wheel selection method, even though there are other available methods that have been studied by other researchers. The proposed method improves the selection chance of each individual and creates an even playing field. This is achieved by making each individual's roulette wheel different from the ordinary method that uses one common roulette wheel for all individuals. The individual roulette wheels contain $s_i$ permission space and $1 - s_i$ non-permission space, and these factors help determine the choices. The process is repeated until the number of selected individuals reaches $n_0$. By selecting one individual and checking its individual roulette test, the selection rate of each individual appears as follows:

$$s_i = \frac{s f_j}{\max_{j=1,2,...,n} s f_j},$$

where $s f_j$ is the fitness value shared by a speciated GA. In this formula, the $s_i$ value is larger than the ordinary rate and reaches 1 when the owner achieves the best fitness value in the generation, i.e., the individual that is selected inevitably. Although ordinary roulette wheel selection contains only one step determined by fitness value, the proposed selection process contains two steps; the individual-taking step and the selection step, and the chromosomes are selected evenly by the equal taking rate of the first step.

Fig. 12 shows the experimental results with the roulette wheel selection (RWS) method and the distributed roulette wheel selection (DRWS) method. The results show that the proposed selection method has similar evolvability (from the fitness value transitions) and scatters selected population (from the Hamming distance transitions).

### 3.6. Genetic operators

For genetic operation we followed the process shown in Fig. 13. We used three kinds of crossover operators and two kinds of mutation operators.

The selection was applied to all individuals of the given population at least once, and then the selection ratio ($s_i$) was considered to determine whether to use that individual or not. With the selection method, the individual with the best-shared fitness was always selected, although all the other individuals had at least one chance of being selected.

We used three crossover methods equally in the experiment, known as one point crossover, uniform crossover and block-unit uniform crossover. The one point crossover method performed crossover at a random point. The uniform crossover method randomly confirmed the performing of crossover between each bit of two chromosomes. This helped in the process of fast gene mixing.

The block-unit uniform crossover method refers to modified uniform crossover, which swaps blocks instead of bits. A chromosome has eight product-terms and eight product-term-disable bits, each with independent information. The block-unit uniform crossover method is concerned with the information. An example of the crossover method is shown in Fig. 14.

We performed two mutation methods equally in the experiments. The first method was a general bit-flip method, which flips the gene bit with regard to the mutation rate. The second method was block-unit mutation, which flips a product-term-disable bit with or without resetting the product-term bits (the reset rate for product-term bit was 50%).

### 3.7. Evolving hardware satisfying strict and complex test

When an EHW have to pass a rigorous test, it is mostly impossible to use the test function as fitness evaluation function. In this case, GA repetition method can be used to make candidates. However, the candidates obtained from the repeated evolution may be redundant; some tests may be inefficient. Speciation can be

![Fig. 12. The transition of fitness value and average Hamming distance for RWS and DRWS (10 trials, simple GA with elitism).](image)

![Fig. 13. The genetic operation process.](image)

![Fig. 14. The uniform crossover method in the unit of the product-term block.](image)
the solution because the speciated hardware ensemble is diverse and characteristic intrinsically. Fig. 15 shows the evolving process for rigorous test with speciated GA and simple GA.

4. Experimental results

In order to investigate the usability of the proposed method, we performed some experiments using simplified GAL circuit simulation. The goal hardware was defined as a 6-multiplexer. This hardware was formerly used for testing other GAL hardwares [6]. The circuit contained four input bits, two selection parameter bits and one output bit.

4.1. Speciated evolvable hardware

All experiments were performed with the experimental parameters shown in Table 2. The parameter values were chosen empirically and we used the distributed roulette wheel selection method with the elite preserving strategy. Among the parameters, the goal solution-search rate refers to a stop condition for the speciation method. An evolution is stopped when the ratio of solutions (circuits) to population reaches the goal solution-search rate.

We observed the capacity of searching multiple solutions. Fig. 16 shows a fitness transition of the number of solutions. The numbers increase steadily after one solution is found.

We observed the degree of fitness sharing and the average fitness values were under the raw fitness values and more variable. This means fitness sharing and speciation operated well. We also observed the transition of the maximum fitness values during the generations in Fig. 17. The first solution appeared at the 171st generation. The number of solutions was satisfied with the goal solution-search rate at the 593rd generation.

As a result of the evolution process used in the above experiment, we obtained 13 chromosomes plus 2 redundant ones (the number of goal solution-searches was 15). Although all 13 of these chromosomes had different structures, they also displayed the same hardware function, i.e., that of the 6-multiplexer. To analyze the difference between the solutions, we conducted single linkage clustering and depicted their dendrogram [19]. There are two dendrograms shown in Fig. 18. The left one was obtained from 15 solutions that were evolved by a conventional GA, and the other was obtained from the 13 solutions that were evolved by the speciated GA. The dissimilarities shown when using the speciated GA are more distinct than those of the conventional GA.

The distance values between the solutions of the speciated GA are distributed from 3.3 to 6.7. Because the distance is 2.4 when the block is completely different, and the distance is 8.5 when two blocks are completely different, all the solutions show at least one-block-complete-difference and at most two-block-complete-difference. In the conventional GA, meanwhile, only the individual 10 shows a distance of two or more. This means they have no diversity.

In order to select representatives and analyze them, we divided the solutions into four groups at the distance point 6, at which (see Fig. 18) the cut-line is depicted by a dotted line. Then, we selected four circuits that represent the groups and drew the circuits in Fig. 19. Each circuit has not only a characteristic structure but also the same substructures that perform the same function. For instance, the AND gates from the 2nd to the 5th part of individual 6 are the same as all the AND gates of individual 1.

In Fig. 20, this is demonstrated well. The gates with the exception of the same part are extra functional parts of the circuit, i.e., individual 0 has an extra function of $\overline{ACDEF}$. Through the analysis of these parts, we may find useful extra functions for each of them. In this experiment, the extra parts are trivial. However, they may display extra strong functions in a large hardware device. This research aims to obtain creative and novel hardware designs.

Obtaining diverse circuits has another benefit, which is that we can find better hardware in any criteria. For example, individual 1 uses the fewest gates and input signals, as shown in Table 3. This

![Fig. 15. Evolving process for rigorous test.](image_url)

![Fig. 16. A case of fitness transition of the number of solutions during the generations.](image_url)

![Fig. 17. The transition of the maximum-shared fitness.](image_url)
means that the circuit 1 is the most simplified structure, and we can select it as the optimized one. In other cases, to obtain a circuit that overcomes hard benchmarks, it is possible to test the circuits after evolution instead of testing during the evolution. This process also saves much time.

There is another benefit of evolving diverse species. When the first entrance of a logic cell is out of order, circuit 1 cannot work well. This means that we can replace it with circuit 2, which works well without the first AND-term, if we have the circuit species. Diverse circuits can be diverse solutions to a problem.

**Fig. 18.** Dendrograms of evolved circuits with single linkage clustering. (a) Conventional GA; (b) speciated GA.

**Fig. 19.** Circuit structures of representatives of the species. (a) Individual 0 in the group 1; (b) individual 6 in the group 2; (c) individual 1 in the group 3; (d) individual 2 in the group 4.
1. A 3-input AND gate causes an error with a rate of 0.00001 at the positions of or1, or2, or3.

2. A 4-input AND gate causes an error with a rate of 0.00003 at the positions of or4, or5, or6.

3. A 5-input AND gate causes an error with a rate of 0.00005 at the positions of or5, or6, or7, or8, or9.

4. A 6-input AND gate causes an error with a rate of 0.00007 at the positions of or5, or6, or7, or8, or9, or10.

Table 3: Comparison of circuit structures (smaller numbers are better, bold characters indicate the best)

<table>
<thead>
<tr>
<th>Circuit Type</th>
<th>Individual 0</th>
<th>Individual 6</th>
<th>Individual 1</th>
<th>Individual 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of gate inputs</td>
<td>17</td>
<td>17</td>
<td>12</td>
<td>18</td>
</tr>
<tr>
<td>Number of used inputs</td>
<td>9</td>
<td>6</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

4.2. Search efficiency comparison

Next, the speed of finding solutions with the speciated GA was compared to that of the conventional GA. Because the speciation algorithm keeps diversity well, it is favorable to searching solutions. We conducted 50 runs, which were performed until the first solution was found. The result is as shown in Table 4.

Table 4: Performance comparison until the first solution is found

<table>
<thead>
<tr>
<th>Hardware</th>
<th>6-MUX</th>
<th>Module A</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA</td>
<td>Speciated GA</td>
<td>Ordinary GA</td>
</tr>
<tr>
<td>Result of seek (seek/trials)</td>
<td>50/50 = 1.00</td>
<td>31/50 = 0.62</td>
</tr>
<tr>
<td>Fastest case (gen)</td>
<td>89</td>
<td>2207</td>
</tr>
<tr>
<td>Slowest case (gen)</td>
<td>573</td>
<td>9999</td>
</tr>
<tr>
<td>Average generation (gen)</td>
<td>249.4</td>
<td>&gt;7859.1</td>
</tr>
<tr>
<td>Standard deviation (gen)</td>
<td>102.8</td>
<td>&gt;2365.6</td>
</tr>
<tr>
<td>Process time (ms)</td>
<td>165,662</td>
<td>&gt;707,672</td>
</tr>
<tr>
<td>Running speed (gen/ms)</td>
<td>0.075</td>
<td>0.555</td>
</tr>
<tr>
<td>Seek success rate per minute</td>
<td>18.04</td>
<td>&lt;2.62</td>
</tr>
</tbody>
</table>

Generation area: 0–9999. Bold characters indicate the better.

4.3. Diversity and usability of the speciated EHW

In order to confirm the usability and the diversity of the circuits obtained from the speciation algorithm, we tested the EHWs. The test was conducted on the circuits obtained from the evolution experiments shown in Sections 4.1 and 4.2, and we supposed that all the circuits had error rates such as the values shown in Table 6.

Table 6: Proposed rules for testing a digital circuit in extreme situations

- Results are shown after using a circuit with multi-input AND gates and an 8-input OR gate (shown in Fig. 19) after 10 h at extreme frequency.

1. A 3-input AND gate causes an error with a rate of 0.00001 at the positions of or1 and or3.
2. A 4-input AND gate causes an error with a rate of 0.00003 at the positions of or4, or5, or6.
3. A 5-input AND gate causes an error with a rate of 0.00005 at the positions of or5, or6, or7, or8, or9.
4. A 6-input AND gate causes an error with a rate of 0.00007 at the positions of or5, or6, or7, or8, or9, and or10.
5. The 8-input-positions of an 8-input OR gate are or1, or2, or3, or4, or5, or6, or7, or8 in order.
Average evolution time (ms) 8775
Number of tested circuits 50 766 1:9.3

Expected seek time for non-error order with regard to the error values. From Fig. 21, it is obvious that of the 15 speciated circuits. The graph shows the sorted results in method. Finally, ‘speciated GA_avg’ marks the average error values after testing the 15 individuals that were evolved by the proposed ‘speciated GA’ means the extreme test results of the best circuits 15-or-more-times evolutions to obtain similar results to this because one-step evolution is not enough to obtain diverse circuits and also, the evolution results can be redundant. The ‘speciated GA_avg’ showed similar results to the simple GA, so we can see that each speciated individual does not show a large difference with individuals that evolved by the simple GA.

Table 7 shows the analysis results obtained in extreme situation tests. In the experiment, the number of evolved non-error circuits was 2 for the simple GA and 9 for the speciated GA. Compared to the number of circuits (766) from the speciated GA in the 50-times tests, the number from the simple GA was only 50. In Table 7, the speed of the simple GA was also worse than that of the speciated GA since the seeking times were 8775 s for the simple GA and 3313 s for the speciated GA. After calculating the expected seeking time when finding the rate of the non-error circuit, the speed of the speciated GA was better, since the expected time of 18,407 is about 12 times smaller than time 219,378 of the simple GA. If we consider the extreme test time, that is 10 h, then the efficiency of the proposed method is much better.

5. Concluding remarks

In this paper, we confirmed that the fitness landscape of EHW is complex and deceptive. The speciation technique and the distributed roulette wheel selection methods were found to be useful. When using the speciation technique, we obtained diverse hardware modules with only one-step evolution. Contrary to the conventional evolution, we did not have to repeat the evolutionary process with different conditions to obtain diverse hardware modules. We also found that the availability of superior extra hardwares as well as preparations for fault situations. In the experiments, the expected evolution time for searching the non-error circuits with the speciation technique was about 12 times less than the time with the ordinary GA.

We confirmed the superior capacity of the distributed roulette wheel selection operator when using the speciated EHW. We think the selection method is useful to EHW. It is necessary to study and compare the method with other methods in the future.

Acknowledgements

This research was supported by the MKE (Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment), IITA-2008-(C1090-0801-0011).

References