Automatic Test Pattern Generation Based on Shuffled Frog Leaping Algorithm for Sequential Circuits

Aijun Zhu\textsuperscript{a*,b}, Li Zhi\textsuperscript{b},

\textsuperscript{a}School of Electronic Engineering and Automation, Guilin University of Electronic Technology, Guilin 541004, China
\textsuperscript{b}School of Mechano-Electronic Engineering, Xi\textsuperscript{d}ian University, Xian 710071, China

Abstract

This paper presents a new approach to the automatic test pattern generation for sequential circuits. According to the structural characteristics of sequential circuits, the expression of Shuffled Frog Leaping Algorithm (SFLA) is constructed and the discrete SFLA model is designed for the automatic test generation. The best test vector set is obtained quickly through the cooperation and competition among the frog group. Finally, experimental results for the international benchmark circuits prove that the proposed algorithm can achieve higher fault coverage and more compact test set when it is compared with similar algorithms.

© 2011 Published by Elsevier Ltd. Open access under CC BY-NC-ND license.

Keywords: shuffled frog leaping algorithm, automatic test pattern generation, sequential circuits;

1. Introduce

Shuffled Frog Leaping Algorithm is a kind of evolutionary computation method based on swarm intelligence. Eusuff and Lansey proposed the algorithm \cite{2} in 2001, which is inspired by the frog prey behavior. Shuffled Frog Leaping Algorithm is similar to Memetic Algorithm, which is based on group cooperative search \cite{1}. At the same time, it is also provided with the advantage similar to Particle Swarm Optimization Algorithm. Shuffled Frog Leaping Algorithm has a lot of strong points, such as few parameters, easy to implement and fast convergence \cite{3}.

* Corresponding author. Tel.: +086-0773-5837285. 
E-mail address: zbluebird@guet.edu.cn.

1877-7058 © 2011 Published by Elsevier Ltd. Open access under CC BY-NC-ND license. 
doi:10.1016/j.proeng.2012.01.054
Shuffled Frog Leaping Algorithm was applied by all kinds of intelligent optimization systems. For example, Mgmt made use of the Shuffled Frog Leaping Algorithm in the water distribution optimization system in 2003 and Alireza applied the SFL algorithm in the mixed linear model series in 2007. In this paper, Shuffled Frog Leaping Algorithm is applied to generate automatic test pattern for sequential circuits \cite{4}.

There are mainly three types of test generation method for integrated circuits. The first type is a deterministic generation algorithm; the second type is a kind of algorithm based on symbols and state tables; the third type is a simulation-based test generation algorithm. The simulation-based test generation algorithm can be used for forward treatment, which can eliminate the complexity of a system. Furthermore, it is relatively simple for complex components. In this paper, the simulation-based test generation algorithm is adopted. There are many successful cases of application based on swarm intelligence, which prove that it is effective. The updating of frog individual location can describe the order of test vector for sequential circuits; the characteristics of frog individual’s mutual cooperation and competition can reflect the internal structure and fault detection for the sequential circuits. Therefore, Shuffled Frog Leaping Algorithm model is designed for the efficient and rapid test generation for sequential circuits in this paper \cite{5}.

2. Shuffled Frog Leaping Algorithm

Shuffled Frog leaping algorithm is based on memetic evolution which can exchange the global information in the group. Each frog is on behalf of a candidate solution in the solution space. When the optimal solution is searched in the solution space of frog group, the advantages and disadvantages are measured by the fitness function. In the continuous space coordinates system of solution, shuffled frog leaping algorithm can be described as following mathematical model. Let the size of frog group P is T, P can be described as the form $P = \{X_1, X_2, \ldots, X_T\}$. The frog of order K in the D dimensional space coordinates can be described as the form $X_k = (X_{k1}, X_{k2}, \ldots, X_{kD})$.

First, all the frogs are randomly initialized, and then the fitness value of each frog are calculated. According to the fitness value of each frog, the frog group can be divided into M subgroups. The size of each subgroup is N, and the relationship of T and N and M can be described as the form $T = M \times N$.

The method for the frog group divided into M subgroups is as follows. According to the fitness value of each individual, the frog can be ordered in descending order. Next, the frog group is divided into M subgroups ($Z^1, Z^2, \ldots, Z^M$). The ith subgroup can be described as the form $Z_i = \{X_i \in P | 1 \leq i \leq N\}$, $1 \leq i \leq M$.

Memetic evolution is carried in each subgroup of the frog for the local optimal search, and then all the subgroups of the frog are mixed together. According to the fitness value of each individual, the frog can be ordered in descending order and divided into new subgroups once more. The loop stops until the fitness value of individual in the group cannot improve any more, or until the number of iterations has reached the maximum limit allowed. Finally, the best frog is selected, which means the global optimal solution is obtained. The method above is the global search strategy for Shuffled Frog Leaping algorithm.

The local optimal search strategy can be described as: the best and the worst frog are found in the light of fitness value, and then the position of the worst frog is renewed according to the following strategy \cite{2}.

$$D_i = \text{Rand} \times (P_b - P_w)$$ (1)

In the formula above, $P_b$ is the position of the best frog whose fitness is the highest, and $P_w$ the position of the worst frog whose fitness is the lowest, and Rand is a random number between zero and one. $D_i$ is between $-D_{\text{max}}$ and $+D_{\text{max}}$, and $D_{\text{max}}$ is the maximum updating step size. If the fitness value of new position is better than the fitness of original position, then the original position is replaced by the new
position. Otherwise, a new frog is produced randomly to replace the original frog. The local search loop does not stop until the number of iterations reaches maximum value allowed, and then the system returns to the global search.

3. Automatic Test Generation Model Based on Shuffled Frog Leaping Algorithm

3.1. Algorithm description

In this paper, only the stuck-at fault is to be considered to simplify the test model. Generally speaking, a test vector consists of “1” and “0” binary series which is determined by the primary input of a circuit. The main task of automatic test generation is to find test vectors which can detect one or more circuit faults, and then apply the test vector on the input node as input stimuli. Finally, the test results are obtained through the fault simulation.

In a sequential circuit, the current input test vector may have some impact on the next primary input, through the model of shuffled frog leaping, so the current best test vector and historical best test vector are saved to generate the following test vector. To establish the shuffled frog leaping model for the automatic test generation, several related definitions are given below.

Definition 1: The D bits of binary string from the Primary Input are defined as frog’s position in D-dimensional space coordinates.

Definition 2: The circuit fault is defined as the food of a frog, and the number of the food is defined as the fitness value in the objective function.

Each bit in the test vector which consists of binary string can be adjusted by the updating of frog’s position. The adjustment above leads to the change of test vector, and then the corresponding test results are obtained. Finally, the best solution is obtained by the evaluation of objective function.

The shuffled frog leaping algorithm is redefined as formula (2).

\[ P_w(t+1) = P_w(t) + R * (P_b(t) - P_w(t)) \] (2)

In the formula above, \( P_b \) is the N-dimensional test vector in the subgroup whose fitness value is the highest, and \( P_w \) is the N-dimensional test vector in the subgroup whose fitness value is the lowest, and R is a random N-dimensional vector, each dimension of which is selected randomly as zero or one. The \( t \) represents the current time, and \( t+1 \) represents the next time.

The operators in the formula (2) can not be regarded as the usual numerical calculation operators. Each operator in the formula is redefined as its new form. Subtraction (-) operator is redefined as follows: During the process of subtraction, the same dimension of the best frog and the worst frog is compared, if they are equal, the value of the dimension is set zero; if not, it keeps the value of the best frog. Multiplication (*) operator is redefined as follows: When the random vector, which is composed of zero and one randomly, are multiplied with the result of subtraction, if the random number of one dimension in a vector is one, then the dimension of multiplication result keeps the value of subtraction difference; if not, it is assigned the value 0. Add (+) operator is redefined as follows: When the worst frog adds the position difference, the value at the same dimension is compared; if they are not equal, then the result at the same dimension keeps the original value; otherwise, it is assigned the value 0.

3.2. Parameters Selection

When the algorithm is designed, the following parameters should be considered: population size, number of iterations, maximum of step length and test sequence length.
The first parameter needed to determine is the population size, which depends on the test sequence length, timing depth and the number of primary input PI. If the test sequence is longer, larger population size will be needed to maintain the diversity of the group. In this paper, if the test sequence is less than 5, then the population size is set as 16; if the test sequence is longer than or equal to 5, the selected population size is 32.

The second parameter is the number of iterations. Only when the number of iterations is large enough, can the best solution be found with great ease. However, if the number of iterations is too large, it will waste a lot of CPU time. Therefore, this paper takes a moderate value. The number of iterations is selected as 12 at local optimization.

The third parameter is the maximum of one step. Step length determines the local search capability. Generally speaking, the smaller the step length, the stronger the local search capability is. However, it will cost longer time. This article sets the maximum step length as the simultaneous changes which two dimensions of position difference vector produce.

The fourth parameter is the test sequence length. The unpredictable faults need a long test sequence to activate and propagate. Test sequence length can be expressed as the product of a sequential depth and the number of primary input PI. The sequential depth refers to the least number of trigger between PI and the longest distance gate.

4. Experimental Results

Table 1 shows the results of the test generation, including the total number of fault in a variety of standard sequential circuits, the number of detected faults and the total length of test vector set. In the table, Det indicates the number of detected faults, and Vec indicates the length of test vector set. The test simulation is based on international standard iscas’89 circuits and several other standard circuits. In order to facilitate comparison, the table presents the traditional test generation results based on genetic algorithm, with the relatively improved magnitude of the fault coverage.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Total number of faults</th>
<th>GA(6)</th>
<th>SFL</th>
<th>Improvement of fault coverage %</th>
</tr>
</thead>
<tbody>
<tr>
<td>s382</td>
<td>399</td>
<td>347</td>
<td>281</td>
<td>358</td>
</tr>
<tr>
<td>s298</td>
<td>308</td>
<td>264</td>
<td>161</td>
<td>273</td>
</tr>
<tr>
<td>s526</td>
<td>555</td>
<td>417</td>
<td>281</td>
<td>419</td>
</tr>
<tr>
<td>s641</td>
<td>467</td>
<td>404</td>
<td>139</td>
<td>408</td>
</tr>
<tr>
<td>s820</td>
<td>850</td>
<td>517</td>
<td>146</td>
<td>534</td>
</tr>
<tr>
<td>s1196</td>
<td>1242</td>
<td>1232</td>
<td>347</td>
<td>1230</td>
</tr>
<tr>
<td>s1423</td>
<td>1515</td>
<td>1222</td>
<td>663</td>
<td>1285</td>
</tr>
<tr>
<td>s1494</td>
<td>1506</td>
<td>1416</td>
<td>245</td>
<td>1475</td>
</tr>
<tr>
<td>am2910</td>
<td>2391</td>
<td>2163</td>
<td>745</td>
<td>2293</td>
</tr>
<tr>
<td>div16</td>
<td>2147</td>
<td>1739</td>
<td>634</td>
<td>1778</td>
</tr>
</tbody>
</table>

From the test results, it can be concluded that the fault coverage based on shuffled frog leap algorithm is mostly higher than that based on genetic algorithm, with only one exception. As the fault coverage in the table is not equal, so the length of each set of test vectors can not be compared directly. It is obvious
that there are some direct relationships between the length of test set and the fault coverage; that is to say, when the higher coverage is wanted, the larger test set is needed.

Because of the different test environments of each algorithm, their running time can not be compared directly. Generally speaking, test generation time is mostly consumed in the fault simulation. If the time of a fault simulation is $t$, then the maximum of total test generation time probably equals to the product of population size, number of iterations, sequential depth, number of optimization and the time of a fault simulation.

5. Conclusions

By the competition and cooperation of the swarm, the test generation results based on shuffled frog leaping algorithm are better than those based on genetic algorithm; that is to say, the higher fault coverage is obtained, and the smaller test set is needed mostly. It is proved that this algorithm is effective. Of course, this algorithm is also easy to fall into the local optimization, so further research is needed to improve the efficiency \cite{7,8} of test generation.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant NO. 60266001 and 60766001.

References

\cite{1} Wang Yamin. Scheduling optimization of assembly line based on shuffled frog leaping algorithm. Computer Engineering and Application, 2010;46(17):52-56.


