Non-persistent Elitism Compact Genetic Algorithm with Tendency and its Application in Evolvable Hardware

JIANWEI MI XIAOLI FANG LIBIN FAN
Key Laboratory of Electronic Equipment Structure Design, Ministry of Education of China
Xidian University
No.2 South Taibai Road, Xi'an, Shaanxi 710071
CHINA
jackiemi@126.com

Abstract: - The compact genetic algorithm uses the value of probability variable to represent the population, and each generation produces two chromosomes by the value. This dominant property makes it efficient to be applied in evolvable hardware, and significantly reduces the storage capacity. However, in case of dealing with the complex issues, its execution effects often fail to reach the required demands. Taking the problems above into account, so this paper presents an improved compact genetic algorithm named None-persistent Elitism TCGA (ne-TCGA), combined with the analysis of convergent trend. On the basis of TCGA (Compact Genetic Algorithm with Tendency), it adopts the strategy of non-durable elitist preservation, which both ensured the adequate selection pressure, and maintained the diversity of the population in the evolutionary process. The analysis shows that the ne-TCGA applied in evolvable hardware has better computational efficiency than other random search algorithm.

Key-Words: - Evolvable hardware, genetic algorithm, ne-TCGA

1 Introduction

After the fixed functional hardware and the reconfigurable hardware, the next generation will be self-configurable and evolvable hardware[1]. They are endowed to solve large-scale and complex problems by using the development patterns of biology[2]. The configuration of logic circuits in evolvable hardware is evolved via the evolutionary algorithm, so the evolvable result and efficiency of the evolutionary algorithm have an crucial influence on whether the evolvable hardware can achieve the desired function or not. At present, the evolvable hardware technology can only accomplish simple small-scale circuits. So it is still facing many problems in the process of development, such as the low speed of evolution, the low efficiency of evolution, the weak robustness of evolvable circuit[3,4]. In order to tackle with these problems, we need to analyze the basic theoretical knowledge thoroughly. It is of particularly importance to investigate an algorithm with favorable effect of convergence and high efficiency of execution.

Therefore, this paper proposes an improved Genetic Algorithm with trend named ne-TCGA (none-persistent elitism TCGA), which improves the convergent efficiency of algorithm and achieves satisfactory performance of application in hardware evolution.

2 Evolvable Hardware

The idea of evolvable hardware came from the 1950's, the father of computer John Von Neumann proposed the vision of developing a machine with self-reproduction and self-healing abilities. EHW develops rapidly. And now it has become one of the popular topics in international academia. Moreover, reconfigurable hardware can be an experiment platform applied to smart analog technology, such as controller of analog neural network[5]. In China, Kang et al early carried out the research about theory and technology of evolutionary hardware[6].

Evolvable hardware refers to using the evolutionary algorithm for configuring the dynamically configurable circuits in programmable logic device and finally evolving into the required logic circuits. Evolvable hardware can change its architecture dynamically like creatures as external environment changes in order to achieve self-organizing, self-adaptive and self-healing abilities. It has a significant meaning for studying evolvable hardware. Since the complexity of hardware systems is increasingly raised, the difficulty in system design is increased and the reliability is on the contrary decreased. It can meet the hardware adaptability to environment by using evolvable

hardware，which makes the system automatically adjust its internal structures in real time for adapting changes of internal and external environment[7]. Thus the system can operate by self-fault-tolerant. Evolvable hardware technology will also have infinite potential in embedded system design, and it can be a promising solution for partitioning hardware and software in embedded system design. In sum, evolvable hardware has a broad application prospect and considerable industrial, commercial value in circuit design, fault-tolerant systems, pattern recognition, artificial intelligence and other fields.

3 The basic idea of ne-TCGA

Genetic Algorithm (GA) is a random search algorithm for the simulation of natural evolution in biosphere by using a series of encoded bit string to describe the population of candidate solutions of the problem. However, because it needs to store large amounts of individual information in a population, occupies a lot of storage space and has huge calculation in dealing with complex issues[8]. Additionally, there are many ways and technologies to improve the application of GA in FPGA[9]. The compact genetic algorithm (CGA, Compact Genetic Algorithm) was a successful one among them [10]. The compact genetic algorithm uses the probability variables to describe the population. Its evolutionary process is that each generation can randomly produce two mutually independent chromosomes according to the probability variables and calculate fitness for them. The chromosome with good fitness value will be treated as "Victor" by comparing their fitness values. Then, it is needed to update variable values of probability corresponding to every individual chromosome according to the obtained "victor" bit string. The evolution process ends and makes the obtained probability variables as an optimal solution for problems until every bit of probability variables is converged to "0" or "1". The bit numbers of probability variable are the same as chromosome, and its value of each bit is the probability of chromosome of which corresponding bit is “1”. Therefore, CGA is very effective in application of limited memory, such as evolvable hardware[9].

3.1 The propose of ne-TCGA

Although CGA is more successful than GA in terms of storage, it has an explicit termination criterion. When every probability variables converges to "0" or "1", the evolution ends. However, because the CGA can achieve less information in the evaluation of chromosome[11] and have poor search ability and insufficient extraction capability to information of excellent individual, it is straightforward to lose the excellent individual and premature to converge, and it can only be used to solve a simple problem of first order. The implementation effect often are not able to meet the actual application requirements for complex issues and the implementation speed is slow. So there is TCGA, TCGA increases judgement for trend from the current solution toward the optimal solution in the algorithm, and introduces the strategy of elitist preservation. But excessive elitist preservation may lead to premature convergence. High selection pressure makes the group converge rapidly. Thus it falls into local optima at the expense of diversity of the population. So this paper proposed ne-TCGA, and introduced the non-durable strategy of elitist preservation and parameters α which represents the maximum algebra of elitist preservation. The elite individuals can be inherited to the next generation within α-generations, but it will regenerate two chromosomes by the probability variables more than α-generations. The following sections will describe the detailed process of ne-TCGA.

3.2 Specific process of ne-TCGA

1) To construct a probability vector P which has the same encoding length L as chromosome, and each bit of probability vector is 0.5. And then the probability vector randomly generates two chromosomes. Each bit value of the probability vector represents the probability of the generated chromosome of which corresponding bit is “1”.

2) To calculate separately the fitness value of two generated chromosomes and compare the two fitness values. The chromosome with big fitness value is treated as "winner". On the contrary, the chromosome with small fitness value is treated as "loser". Then comparing each bit of the "winner" and "loser", if the two corresponding numbers are not equal, then proceed 3). Otherwise, continue to compare the next bit.

3) To invert each bit of the "winner", and then, to judge and compare each fitness value between the inverted individual and the original individual. If the inverted fitness increases, to judge the value of inverted bit. If its value is “1”, to update

corresponding bit value of probability variable by adding the 1/N step length. If its value is “0”, to update corresponding bit value of probability variable by reducing the 1/N step length. The N affects the computational cost and storage in the algorithm running process, so it is defined as the population quantity.

4) To determine whether it reaches the convergent conditions. If it reached, then end the process. If it didn’t, then proceed to the next step.

5) There are chromosome mutation operations in this step. To judge whether each bit of probability variables is bigger than 0.5, if it’s bigger than 0.5, then continue to determine the corresponding bit of the "winner" chromosome. If its value is “1”, then to remain constant. If its value is “0”, then to invert the bit; Otherwise, if the corresponding bit of "winner" is “1”, then to invert the bit. If it is “0”, then to remain constant. To judge and compare the fitness value of generated chromosome with the fitness value of original "winner" chromosome, and make the chromosome with big fitness value as new "winner."

6) To judge the algebras of elitist preserving. If it’s not more than α, it will generate a new chromosome by the probability vector and turn to 2). If it’s over α, it will generate two new chromosomes by the probability vector and turn to 2).

The pseudo-code of ne-TCGA algorithm is as follows:

```
Step.1  for i = 1 to L  do  P[i]=0.5;
      a=generate(p);
      b=generate(p);
Step.2  if fitness(a)>fitness(b)  then
        winner=a;loser=b
      else  winner=b;loser=a;
Step.3  for i = 1 to L
      if winner[i] != lose[i]  then
       if winner[i]==1  then
       { winner[i]=0;fw=fitness(winner)
        If fw>fwn then P[i]=P[i]-1/N;
        else P[i]=P[i]+1/N;}
       else{ winner[i]=1;fw=fitness(winner)
           if fw>fwn then P[i]= P[i]+1/N;
           else P[i]=P[i]-1/N;}
Step.4   for i = 1 to L do
        if P[i]>0&&P[i]<1  then goto Step.5
        else end all
Step.5   if winner==a then
{c=mutate(a);
        if fitness(c)>fitness(a) then a=c;}
      else{c=mutate(b);
       if fitness(c)>fitness(b) then b=c;}
Step.6   if z<α a=winner;b=generate(P);z++
      else a=generate(P); b=generate(P);z=0;
```

4 Comparison of experimental results of ne-TCGA, CGA and TCGA

Ne-TCGA inherits the advantage of less storage space in CGA, and in which there are judgments for trend from the current solution toward the optimal solution. It can increase the search ability of algorithm and quickly make the chromosome individual converge to optimal solution. Then, in order to get the better chromosome individual with maximum probability, the improved the mutation operation is introduced. Finally, to control the generation of elitist preserving, we introduce the non-durable elitist preserving strategy, so that ensured the adequate selection pressure, and maintained the diversity of the population for the reasonable control of elitist preservation at the same time. It has a great advantage of the control in premature convergence. This paper uses CGA, TCGA, and ne-TCGA to separately seek the maximum of two functions as follows.

$$y = \sin\frac{\pi}{180}x - 5x^2 + 60x + 800 \tag{1}$$

$$y = s\sin 10\pi x + 2 \tag{2}$$

where, x is real independent variable, and y is the dependant value of the functions.

The graph of functions in Eqs.(1) and (2) are shown in Fig.1 and Fig.2, respectively. The result of comparisons on performances and required numbers of evaluations are shown in Figs. 3, 4, 5 and 6.

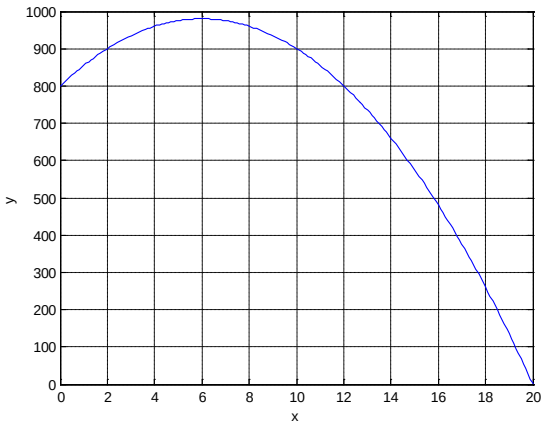

Fig.1 Curve of function in Eq.(1)

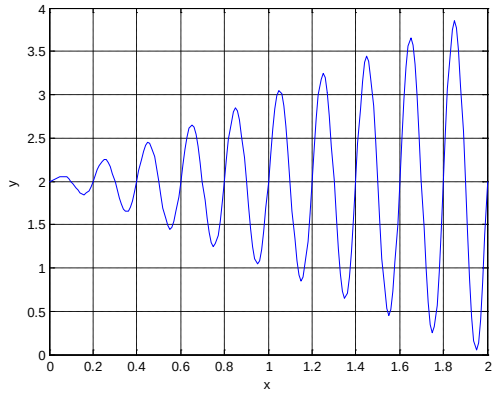

Fig.2 Curve of function in Eq.(2)

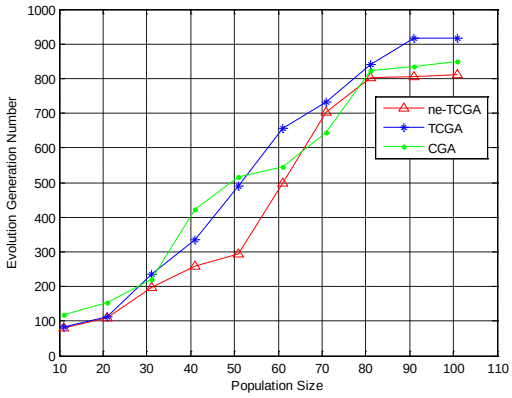

Fig.3 Evolution generation number of function in Eq.(1)

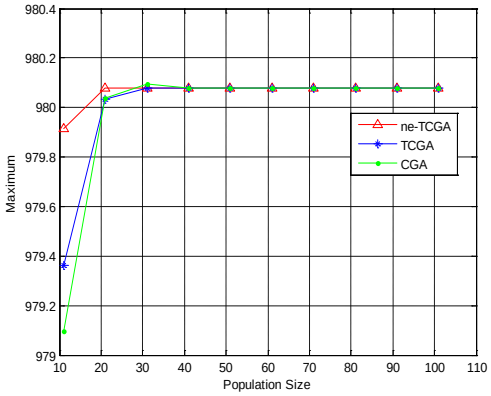

Fig.4 Maximization of function in Eq.(1)

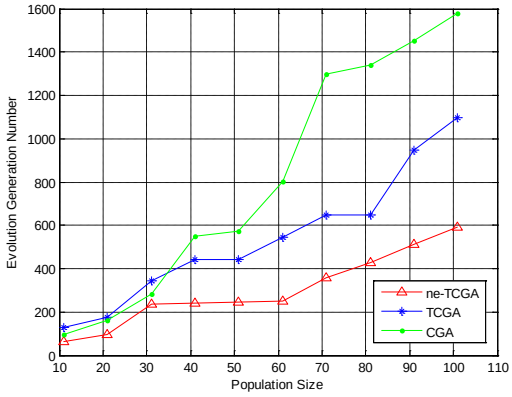

Fig.5 Evolution generation number of function in Eq.(1)

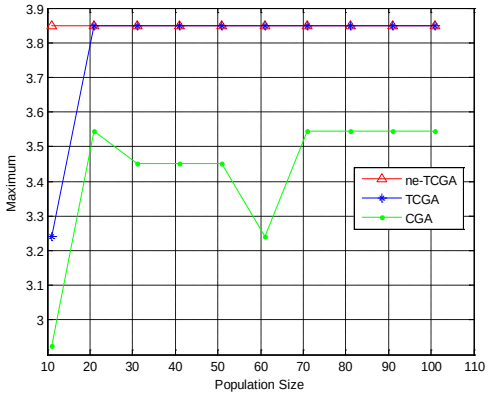

Fig.6 Maximization of function in Eq.(2)

As can be seen from the figures, the evaluation numbers of the three algorithms are increasing with N increases. But the ne-TCGA algorithm always has the minimum number of evaluation and the best performance of solution. When seeking the maximum of the function in Eq.(2), although CGA once searched the maximum in domain of definition in the evolutionary process, the result converged to local maxima and did not get the maximum with poor performance of solution.

5 The application of ne-TCGA in the evolvable hardware

Evolvable hardware is used to encode structures and parameters of circuit which are treated as chromosome and execute evolutionary operation. The classical genetic algorithm occupies large memory and has large calculations for extensive range searching. Therefore, in order to reduce the storage and speed up the efficiency of searching, we introduced the ne-TCGA algorithm proposed previously, This paper uses ne-TCGA algorithm which regards a full adder as the evolutionary target. It chooses FPGA development board which is Altera's Stratix II family EP2S30F484I4 to design self-evolutionary system and verifies the validity of algorithm.

5.1 Hardware design of self-evolutionary System

The designed self-evolutionary system contains the virtual reconfigurable circuit-based IP core. Evolvable IP core is an evolutionary circuit which is controlled by evolutionary algorithm and can run on the Nios II soft-core. The designed IP core in this paper is a Cell Array consisting of cell array of 8 × 5[12], as shown in Fig.7. The input of 8 cells in the first column is made up of the 8 inputs which is

provided externally and the inversion of 8 inputs. The cell input in the second column is constituted by the external 8 inputs and 8 outputs in the first column. The inputs in the third column and the subsequent column are made up of the output of the first two columns. This Cell Array has eight outputs.

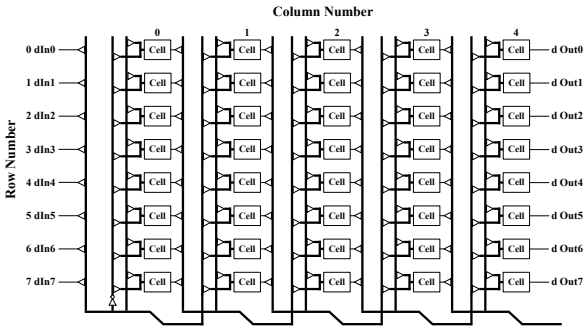

Fig.7 Schematic diagram of the Cell Array

The designed basic logic configuration cell is shown in Fig.8, which is composed of the look-up table and multiplexer. As can be seen from the figure, the cell unit has three inputs and one output. Because the Cell Array has 16 external inputs, the three inputs are determined by three selectors of 16-to-1 and the output of selector can act as control bits to determine the input of look-up table (LUT) behind. Because each selector of 16-to-1 is configured by four chromosomes and (LUT) look-up table needs 8 bits, to configure a single cell totally requires $(3 \times 4) + 8 = 20$ bits. Since Cell Array contains 40 cell units, it needs $20 \times 40 = 800$ chromosome to configure a Cell Array.

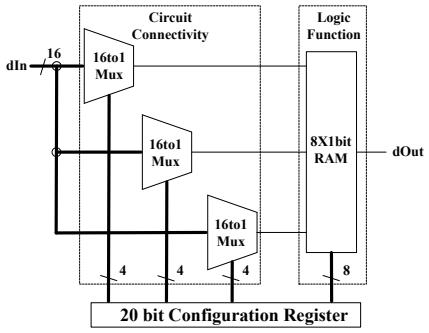

Fig.8 Schematic diagram of internal structure in cell unit

Evolvable IP core has 8 external inputs and 8 outputs, but a full adder only has 3 inputs and 2 outputs. So it just takes 8 combinations of low three places in 8 external inputs as the input of full adder in this paper. This operation will be used in the Nios II IDE by C language. After inputting the test vectors, it operates the outputs of Cell Array XNOR the true table of 8 inputs. The higher 6 bits of the obtained signals are shielded. This operation can be explained by the following reason. The full adder has only 2 outputs, it puts the 0x3 into MASK register and operates AND with matched signals. Then outputs match data and computes "1" numbers of match data, that is the fitness value.

5.2 The verification self-evolutionary system

The ne-TCGA algorithm in this paper is implemented using software method in Nios II soft-core. We download the program to the board by the JTAG download cable to validate the self-evolutionary system. The result shows that although the designed self-evolutionary system using ne-TCGA algorithm has a slight increase in evolution generation than the GA algorithm, it can greatly reduce the storage space.

6 Conclusion

We proposed the ne-TCGA algorithm based on these problems of the Genetic Algorithm applied to evolvable hardware occupies large memory and the CGA algorithm has poor search capabilities when dealing with complex issues. The non-durable elitist preserving strategy is added on the basis of TCGA algorithm, thereby in case of inheriting advantages of TCGA algorithm, such as the less storage space occupied, the definite condition of convergence, the good convergence, the strong search capabilities, it reduces the possibility of falling into local optimal solution and the convergence and search capability are better than TCGA. By applying the ne-TCGA algorithm to the designed self-evolutionary system, we proved the superiority of ne-TCGA algorithm on storage space. The ne-TCGA algorithm will have great potential in hardware evolution by its simple implementation methods and advantage of less storage space occupied in hardware.

Acknowledgement

The authors would like to appreciate the Editor, Associate Editor, and the reviewers for their valuable comments and suggestions.

References:

[1] HARTENSTEIN R. Trends in reconfigurable logic and reconfigurable computing. 9th IEEE International Conference on Electronics. DUBROVNIK, CROATIA, 2002: 801-808.

[2] Gordon TGW , Bentley PJ. Towards development in evolvable hardware.

NASA/DOD Conference on Evolvable Hardware. ALEXANDRIA, VA, 2002:241-250.

[3] Fernando PR, Katkoori S, Keymeulen D, Zebulum R, Stoica A. Customizable FPGA IP core implementation of a general-purpose genetic algorithm engine. IEEE Trans Evol Comput. 2010, 14(1):133–149

[4] Yutana Jewajinda, Prabhas Chongstitvatana. A parallel genetic algorithm for adaptive hardware and its application to ECG signal classification.Neural Comput & Applic. 2013, 22:1609–1626

[5] Amaral JFM , Amaral JLM. Towards evolvable analog airtificial neural networks controllers. 6th NASA/DoD Conference on Evolvable Hardware. Jet Propuls Lab, Seattle,2004:46-52.

[6] Kang Li-Shan, He Wei, Chen Yu-Ping. Evolvable hardware realized with function type programmable device. Chinese Journal of Computers, 1999, 22(7): 781- 784

[7] Zhao Shu-Guang, Liu Gui-Xi. Basic basic theory and key technology of evolutionary hardware. System Engineering and Electronic Technique.2002,24(1):70-73.

[8] Marco A. Moreno-Armendáriz, Nareli Cruz-Cortés,Carlos A. Duchanoy,et al, Hardware implementation of the elitist compact Genetic Algorithm using Cellular Automata pseudo-random number generator. Computers and Electrical Engineering,2013(39):1367-1379.

[9] Fernando PR, Katkoori S, Keymeulen D, Zebulum R, Stoica A.Customizable FPGA IP core implementation of a general-purpose genetic algorithm engine. IEEE Trans Evol Comput,2010,14(1):133–149.

[10] Harik G R. The Compact Genetic Algorithm[J]. IEEE Trans. on Evol. Comput., 1999, 3(4): 287-297.

[11] Cupertino F, Mininno E, Lino E, Naso D. Compact genetic algorithms for the optimization of induction motor cascaded control. Electric machines & drives conference,2007:7–82.

[12] LIU Jieli, YAO Rui. The Implement of Evolvable Hardware on the Design Method of SOPC. Journal of Jiamusi University. 2012,30(2):109-212.