Improved Binary Decision Diagram based Accelerated Circuit Evolutionary Algorithm

Yafeng Meng, Junbin Zhang, Jinyan Cai

Department of Elec. and Opt. Engineering, Mechanical Engineering College, Shijiazhuang, China
myfrad@163.com

Keywords: EHW; EA; Accelerated Circuit Evolution; Binary Decision Diagram

Abstract. Nowadays, Evolvable Hardware (EHW) is widely used in many fields. When it is used in circuit design, its advantages can be incarnated adequately. However, if the circuit scales are very big, the probability of successful circuit evolution is reduced greatly. So improving traditional Evolutionary Algorithm (EA) is very important. In this paper, an improved EA is proposed, and it is combined with Binary Decision Diagram (BDD). Through the simulation experiment, the feasibility and effectiveness of proposed accelerate circuit evolutionary algorithm has been proved.

Introduction

With the development of electronic technology, a novel technique was used in electronic circuit, which name is Evolvable Hardware (EHW). It has many advantages, including self-adaptive, self-repair, self-organization and so on[1-4]. The electronic circuit systems can be improved by it.

EHW's combinatorial optimization and global search tools is EA, in order to obtain the expectable circuit and electronic systems by simulating evolution. The expression of EHW can be expressed as follows, $EHW = PLDs + EAs$ [1, 5]. Evolvable Hardware is abbreviated by EHW, Programmable Logic Device is abbreviated by PLD, and Evolutionary Algorithm is abbreviated by EA. Its basic theory is shown in Fig.1 [5].

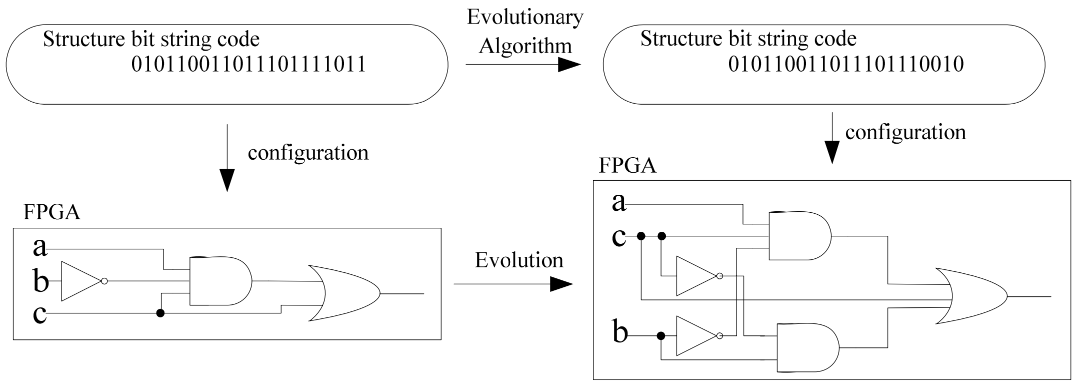

Fig.1 The basic theory of EHW

PLD is the basic hardware of EHW, and EA is the key to determine whether EHW can be implemented or not[5-7].

Many limitations are existed in EA, such as slow convergence speed, etc. So improving EA is very important (including reducing the number of convergence iteration, raising the accuracy of searching and accelerating the convergence speed). If the scales of evolved circuit are very big, the efficiency of evolution circuit will be affected.

In this paper, an accelerated circuit EA based on improved Binary Decision Diagram (BDD) was proposed. The performance of proposed algorithm is validated by the simulation experiment.

The rest of paper is organized as follows. In section 2, accelerated circuit EA based on improved BDD is expatiated. In section 3, the processes of accelerated circuit EA is analyzed. In section 4, the simulation experiment is analyzed. Section 5 is concludes of the paper.

Accelerated circuit EA based on improved BDD

In the process of EA implementation, the evolution of the large-scale circuit will meet with difficulties. The scale of evolved circuit will be the exponential growth following the increased number of the input ports, and the evolution of convergence rate will be the exponential growth following the increase of truth table scale. Therefore, the study of how to accelerate the large-scale truth table circuit will seem especially important.

In order to resolve this problem, divide-and-conquer technique was widely used in EHW. The improved BDD was proposed in this paper. In the process of evolution, the truth table is separated.

When truth table was separated, all inputs could not be separated. Because a same function can be realized by different circuit structure. However, they do not have same hardware resources consumption, and they have different characteristics. Different structures can be used in different environments. Furthermore, if the truth table is separated completely, only a fixed circuit structure can be obtained. If EHW is combined with BDD, not only the circuit structrure can be enriched, but also the speed of evolution circuit can be accelerated.

In order to facilitate the commencement of EA, and make full use of the features of EA, the truth table was splited up only four input ports. The remaining four input ports will be evolved by EHW, the final circuit can be obtained. Furthermore, the another reason of separating remaining four input ports is that FPGA is used as the hardware platform of EHW, and the Look-up Table (LUT) of current mainstream FPGA has four input ports. Although LUT can be considered as a ROM, any four-input one-output truth table can be stored directly[8]. However, EA mainly used to design circuit in this paper.

The basic theory of accelerated circuit EA

Any combinatorial circuit has a truth table, and it can be expressed by a boolean function. We can assume that the boolean function is $f(x)$. First of all, it need to assume that the evolved truth table has M input ports ($x_1,x_2,x_3,\cdots x_M$) and N output ports($y_1,y_2,y_3,\cdots y_N$). $f(x)$ is target function which need to be evolved at the same time, $f_1(x)$ and $f_2(x)$ are assumed unknown boolean function. Then the detailed steps of accelerated circuit EA can be expatiated as follows.

First of all, calculating the number of truth table input ports, it is represented by $innum = M$. The number of truth table output ports also need to be calculated, it is represented by $outnum = N$. Then optionally choose an input port(x_1), and analyze the value of x_1.

Secondly, if there is only "0" or "1" existing in the a kind of value of x_1, in this case, the input port x_1 need to be separated. The remaining truth table is still a whole truth table, and input ports become $innum = M-1$. However, if the value of x_1 is “0”, also can be “1”, in this case, the input port x_1 need to be separated. The remaining truth table becomes two sub-truth table. The input ports of remaining truth table become $innum = M-1$.

Eventually, above step should execute periodically. Until the number of rest inputs ports are

$innum = 4$. Only the remaining four-input ports truth table need to be evolved rapidly at this time.

In the process of circuit evolution, it will need Genetic Algorithm Particle Swarm Optimization (GAPSO) to evolve $f_1(x)$ and $f_2(x)$ [9-13]. The implementation steps of GAPSO will be shown as follows.

Step1 Initializing the particle populations. M chromosome codes need to be generated randomly which meet the requirements. The crossover rate, mutation rate and inertia weight of GAPSO need to be set.

Step2 Calculating the fitness ($fitness_i$) of each particle, and $i < M$. The best particle's position ($gbest_i$) need to be selected of current generation, and compare with the global optimal particle position ($pbest$). Then the optimal position should to be saved. The operations of selection, crossover and mutation are implemented for particle population.

Step3 According to the Equation (1) updates each particle's position ($x_{i,j}^{t+1}$), and according to the Equation (2) updates each particle's speed ($v_{i,j}^{t+1}$).

$$x_{i,j}^{t+1} = x_{i,j}^{t} + v_{i,j}^{t+1} \tag{1}$$

$$v_{i,j}^{t+1} = wv_{i,j}^{t} + c_1 r_1^1 (pbest_{i,j}^{t} - x_{i,j}^{t}) + c_2 r_2^1 (gbest_{i,j}^{t} - x_{i,j}^{t}) \tag{2}$$

Step4 Calculating the fitness value of each particle, and compared with the global optimum value. Then the global optimum value should be updated.

Step5 According to the calculated fitness value to estimate whether the algorithm meets the convergence criteria or not. If convergence criteria is met, the finally particle code can be output. If convergence criteria is not met, Step3 need to be implemented continuously. Then the operations of GAPSO are finished.

In the implementation of PSO, Equation (3) is used as the commonly used fuzzy function for particle discretization[14-16].

$$sig(x_{i,j}) = \frac{1}{1+e^{-x_{i,j}}} \tag{3}$$

While the updating expression of every particle is given as expression (4).

$$X_{i,j}(t+1) = \begin{cases} 0 & rand > sig(X(i,:)) \\ 1 & rand \le sig(X(i,:)) \end{cases} \tag{4}$$

The simulation experiment

In order to validate the effectiveness of proposed accelerated circuit evolution strategy which is based on improved BDD. The typical of C17 is selected as a accelerated evolve circuit. Because of the length of the paper, its truth table is not shown in this paper.

When using improved BDD, only one input port of C17 will be chosen by EA. Its name is assumed X1. Because the value of X1 can be “0”, and can also be “1”. The true table is divided into two parts based on this standard. The rest input ports are still need to separate like the process of separating the X1 input port. If the number of remaining input ports is four, the separating process can be stopped. Because the C17 circuit only has five input ports, the separated process need to be operated once.

In the process of GAPSO, the fitness ($fitvalue$) of algorithm will be expressed as Equation (5).

$$fitvalue = \sum_{i=1}^{2^n} fitnumber_i \tag{5}$$

n s the number of evolved circuit input ports, it is different from the total number of input ports (N). $fitnumber_i$ is the test value of ith input combination ($fitnumber_i \in \{0,1\}$). The fitness function is converted to calculated the maximum of $fitvalue$.

In the process of GAPSO, some parameters need to be set. The specific parameters settings are shown in Table 1.

Table 1 The parameters' value

Operation parameters	The number of particles populations	Crossover rate	Mutation rate	The number of crossover points	The number of mutation points
Value	50	0.90	0.03	2	2

The max number of iterations is 16,000 times. The contrastive curve between traditional EA and improved BDD is shown in Fig.2. They are used in the circuit evolution.

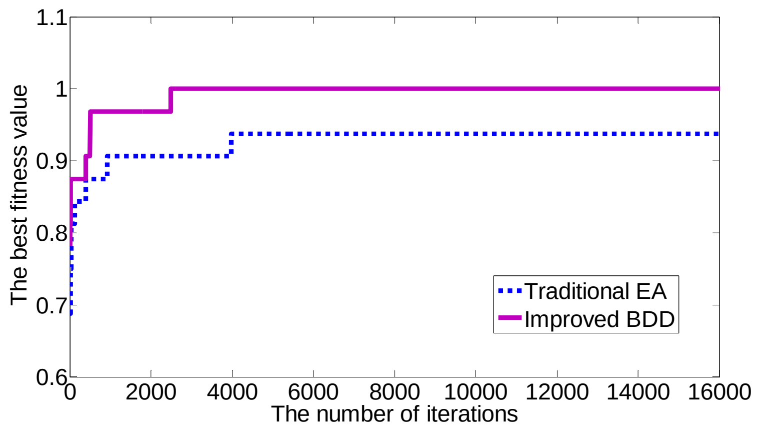

Fig.2 The contrastive curve between traditional EA and improved BDD in EHW

Some conclusions can be obtained from Fig.2 easily. The final fitness value of using traditional EA is 0.9375, and the final fitness value of using improved BDD is 1. However, the best fitness value is 1.

When the algorithms reach the max number of iterations (16,000 times), the traditional EA cannot reach the best fitness. Its fitness value reaches 0.9375 when the number of iterations is 3982. Its fitness value is not change in the future. In other words, the circuit evolution does not achieve success at this time.

The improved BDD can reach the best fitness when the number of iterations is 2493. It only spends 4.625 seconds when it is run in MATLAB. The circuit evolution achieves success.

Through the contrast experiment, when the improved BDD is used in EHW, its performance is better than the traditional EA. The improved BDD can greatly accelerate the evolutionary algorithm convergence rate. The effectiveness of the proposed circuit evolution strategy has also been proved in this paper.

Conclusions

In this paper, the traditional EA was improved by BDD. It is different from the existing improvements of EA. Through the simulation, the feasibility and effectiveness of proposed accelerate circuit evolutionary algorithm has been proved.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (NO.61372039).

References

[1] C.H. Pauline, M.T. Andy. Challenges of evolvable hardware: past, present and the path to a promising future. Genet Program Evolvable Mach. (2011)183-215.

[2] Y.M. Lee, C.S. Choi, S.G. Hwang, et al. Transistor-Level Evolution of Digital Circuits Using a Special Circuit Simulator. Lecture Notes in Computer Science. Vol.5216, No.1, (2008)320-331.

[3] S.Z. Ricardo, S. Adrian, K. Didier, et al Evolvable Hardware System at Extreme Low Temperatures. Lecture Notes in Computer Science. Vol.3637, No.1, (2005)1-37.

[4] D.L. Jason, S.H. Gregory, S.L. Derek. Evolutionary design of an x-band antenna for NASA's space technology 5 mission. Proceeding of the 2003 NASA/DoD Conference on Evolvable Hardware (EH'03), Chicago, USA. 2003, pp.1-9.

[5] J.B. Zhang, J.Y. Cai, Y.F. Meng, et al. Fault self-repair strategy based on evolvable hardware and reparation balance technology. Chinese Journal of Aeronautics, Vol.27, No.5, (2014)1211–1222.

[6] D. Keymeulen, R.S. Zebulum, Y. Jin, et al. Fault tolerant evolvable hardware using field programmable transistor arrays. IEEE Transactions on Reliability. Vol.49, No.3, (2000)305-316.

[7] Y. Moritoshi, H.K. VJung, Y. Ikou. Evolvable Reasoning Hardware: Its Prototyping and Performance Evaluation. Genetic Programming and Evolvable Machines. Vol.2, No.3, (2001)211-230.

[8] U. Andres, S. Eduardo. Evolving Hardware by Dynamically Reconfiguring Xilinx FPGAs. Lecture Notes in Computer Science. Vol.3637, No.1, (2005)56-65.

[9] Y.R. Zhou, Y.X. Li, Y. Wang, et al. Multiobjective Optimization Algorithm Based on (μ+1) Evolutionary Strategy. Computer Engineering, Vol.29, No.18, (2003)1-3.

[10] R. Poli, J. Kennedy, T. Blackwell. Particle swarm optimization. Swarm Intelligence. Vol.1, No.1, (2007)33-57.

[11] E.J. Solteiro Pires, J.A. Tenreiro Machado, P.B. Moura Oliveira. Dynamical modeling of a genetic algorithm. Signal Processing. Vol.86, No.10, (2006)2760-2770.

[12] Y. Jiang, T.S. Hu, C.C. Huang, et al. An improved particle swarm optimization algorithm. Applied Mathematics and Computation. Vol.193, No.1, (2007)231-239.

[13] B. Niu, Y.L. Zhu, X.X. He, et al. MCPSO: A multi-swarm cooperative particle swarm optimizer. Applied Mathematics and Computation. Vol.185, No.2, (2007)1050-1062.

[14] Q. Kanga，H. He. A novel discrete particle swarm optimization algorithm for meta-task assignment in heterogeneous computing systems. Microprocessors and Microsystems. Vol.35, No.1, (2011)10-17.

[15] E.X. Chen，J.Q. Li，X.Y. Liu. In search of the essential binary discrete particle swarm. Applied Soft Computing Journal. Vol.11, No.3, (2011)3260-3269.

[16] Q.K. Pan，M. Fatih，Y.C. Liang. A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. Computers and Operations Research. Vol.35, No.9, (2008)2807-2839.