An evolutionary algorithm based on novel hybrid repair strategy for combinational logic circuits

Xinjie Huang^{a)}, Ning Wu^{b)}, Xiaoqiang Zhang, and Yaoping Liu

College of Electronic and Information Engineering,

Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China

a) huangxj@nuaa.edu.cn

b) wunee@nuaa.edu.cn

Abstract: Evolvable hardware has been applied to the design of combinational logic circuits. But it usually takes too much time using evolutionary algorithm directly, especially designing large scale circuits. In this paper, a novel hybrid repair strategy is proposed, which is a combination of evolutionary repair method and Quine-McCluskey repair technique. Based on the hybrid repair strategy, an evolutionary algorithm is presented for the designs of combinational logic circuits. The experiment results demonstrate that the evolutionary algorithm proposed in this paper can keep balance between the time cost and gates consumption very well, and consumes fewer gates than the previous works.

Keywords: evolvable hardware, evolutionary algorithm, combinational logic circuits, repair technique

Classification: Integrated circuits

References

- [1] T. Higuchi, M. Iwata, D. Keymeulen, H. Sakanashi, M. Murakawa, I. Kajitani, E. Takahashi, K. Toda, N. Salami, N. Kajihara and N. Otsu: *IEEE Trans. Evol. Comput.* **3** (1999) 220. DOI:10.1109/4235.788492
- [2] G. Zbysek and S. Lukas: *Proc. Evolvable Systems: From Biology to Hardware* (2010) 13. DOI:10.1007/978-3-642-15323-5_2
- [3] A. Stoica, R. Zebulum, et al.: *AAAI Spring Symposium Series* (2003).
- [4] E. Stomeo, T. Kalganova and C. Lambert: *IEEE Trans. Syst. Man Cybern. B Cybern.* **36** (2006) 1024. DOI:10.1109/TSMCB.2006.872259
- [5] H. Liang, W. Luo, Z. Li and X. Wang: *Evolvable Systems: From Biology to Hardware* (Springer Berlin Heidelberg, 2010) 193.
- [6] X. Zhang and W. Luo: *2012 IEEE Congress on Evolutionary Computation (CEC)* (2012) 1. DOI:10.1109/CEC.2012.6256165
- [7] X. Huang and N. Wu: *Proc. International MultiConference of Engineers and Computer Scientists 2015* (2015) 674.
- [8] H. Liang: Ph.D. dissertation, School of Computer Science and Technology, University of Science and Technology of China, Hefei, China (2009) (in Chinese).
- [9] L. Sekanina: *Proc. Applications of Evolutionary Computing: EvoWorkshops*

- 2005 (2005) 185. DOI:10.1007/978-3-540-32003-6_19
[10] X. Zhang and W. Luo: 2012 12th International Conference on Hybrid Intelligent Systems (HIS) (2012) 390. DOI:10.1109/HIS.2012.6421366

1 Introduction

Evolvable hardware (EHW) refers to self-reconfiguration hardware design, where the configuration is under the control of an evolutionary algorithm (EA). And the EHW has also been applied to the design of combinational logic circuits nowadays [1]. Compared with traditional methods, the evolutionary method needs not too much specialized experience, but could design creative circuit structures [2].

However, the scalability problem [3], which is one of main problems of the EHW, needs to be solved. As the circuit scale expands, the length of chromosome in EA is increasing, causing significant increase of the time spent. When designing combinational logic circuits with EA, the *stalling effect* [4] is common. As the stalling effect appears, the best fitness of population is improved very slowly, and its tiny increase generally needs numbers of evolutionary generations.

For designing large scale combinational logic circuits, a new EHW algorithm based on generalized disjunction decomposition (GDD) is outlined in [4]. The design process of GDD-based-EHW algorithm can be divided into four steps: firstly, the truth table of target circuit is decomposed to several small parts; secondly, because these small parts have the same inputs, these small parts can be merged into a new truth table; and then, the circuit corresponding to the new truth table is evolved with EA; finally, a multiplexer is used to address the outputs of the evolved circuit for rebuilding the original circuit outputs. Although the GDD is effective to large scale combinational logic circuits, the stalling effect is still ineluctable, and the time spent of EHW algorithm proposed in [4] is still large and can be further improved.

In order to reduce the negative effect of stalling effect, the repair-based-EHW algorithms are proposed in [5], [6] and [7]. The repair-based-EHW algorithms are composed of two steps: firstly, an almost but not complete functional circuit is generated with EA; and then, the uncompleted circuit is repaired with repair method.

In [5], a *direct-repair* technique was proposed by Liang et al., which is the first repair technique applied to evolutionary design of combinational logic circuits. The *direct-repair* technology is based on minsters of the repair truth table to generate repair circuit directly. So the EHW algorithm containing the *direct-repair* technique spends considerably shorter time than traditional EHW algorithm. Nevertheless, the gates consumption in repair circuit is too large.

To further reduce the gates, an *evolutionary-direct-repair* technique was adopted in [6]. The repair technique firstly adopts evolutionary repair method to the repair circuit with a few of gates, and then *direct-repair* method is adopted to continue repairing when the number of the evolutionary repair reaches the upper limit. Compared with using *direct-repair* only, the repair circuits generated by *evolutionary-direct-repair* technique have fewer gates, but the repair time is prolonged and the used gates in repair circuit are still large.

In [7], we also proposed a Quine-McCluskey repair technique (QMRT) to overcome the shortage of *direct-repair* technology. The QMRT takes advantages of Quine-McCluskey algorithm to simplify logic expressions of the repair circuits, thus the gates required in repair circuits are reduced. Although the QMRT spends much fewer gates than the *direct-repair* technique, the repair circuit designed by the QMRT needs more gates as the number of inputs and outputs increases.

In the EHW algorithms proposed in [5, 6, 7], the stalling effect is the only flag to determine that if the repair techniques are started-up. But time consumption of the proposed EHW algorithms increases due to the fact that stalling effect spends much time.

This paper presents a novel hybrid repair strategy (HRS) for the design of combinational logic circuits. The HRS is a combine of evolutionary repair method and the QMRT, and includes *predefined value* as one of flags to trigger or change repair method. The *predefined value* is related to the amount of completed functions directly. The main contributions of this paper are listed in the following.

First, to reduce gates consumption, the proposed HRS adopts evolutionary repair method to repair many functions with a few of gates. Second, in order to consume short time, the QMRT is selected as a necessary repair method in the final stage of HRS to complete the remaining repair circuit. Third, to further shorten time consumption, *predefined value* is introduced to trigger the proposed repair technique and act as a flag of changing repair method.

Based on the HRS, an evolutionary algorithm, named EA-HRS, is proposed. With the HRS, EA-HRS shows a better performance in gates and time consumption compared to the EHW algorithms proposed in [5] and [6], costs fewer gates than the EHW algorithm presented in [7], and spends less time than the EHW algorithms outlined in [4] and [10].

2 The proposed algorithm

In EHW algorithms based on repair technique, firstly, EA is adopted to evolve an uncompleted circuit, and then repair circuit is generated by repair technique. So, the target circuit is composed of the uncompleted circuit and the repair circuit, which are connected with XOR [8] as shown in Fig. 1. It is to be noted here that the repair circuit can also be composed of several circuits which are connected with XOR.

Fig. 1. The structure of target circuit

2.1 An evolutionary algorithm based on a hybrid repair strategy

A novel evolutionary algorithm based on a hybrid repair strategy (EA-HRS) is put forward here. By using EA-HRS, uncompleted circuit is generated by EA and repair circuits are created by HRS.

EA-HRS (<i>Target_Circuit</i> , <i>Stalling_Generation</i> , <i>Threshold_Fitness</i>)	
In: <i>Target_Circuit</i> is the circuit to be designed. <i>Stalling_Generation</i> means the generations used to identify the stalling effect. <i>Threshold_Fitness</i> is the best fitness to trigger the QMRT process.	
Out: An uncompleted circuit expressed by the CGP model.	
1.	Randomly initialize the population with the parameter P (population quantity) and $S_G = 0$.
2.	Evaluate the population according to the truth table of target circuit and evolve these individuals with EA.
3.	IF (the best fitness is larger than <i>Threshold_Fitness</i>) Output the uncompleted circuit, and trigger the HRS process.
4.	ELSE
5.	IF (the best fitness do not change) S_G++ .
6.	IF ($S_G \geq \textit{Stalling_Generation}$) Output the uncompleted circuit, and trigger the HRS process.

Fig. 2. The pseudo code of EA-HRS

The adopted EA is genetic algorithm (GA), which is one of typical EAs, in this paper. The layout of evolved circuit is Cartesian Genetic Programming (CGP) [9] model. The pseudo code of EA-HRS is shown in Fig. 2.

In the EA-HRS, the target circuit is evolved by EA firstly. In the evolutionary process, if the best fitness is larger than *Threshold_Fitness* (in Step3) or the stalling effect occurs (in Step6), the uncompleted circuit is outputted, and the algorithm enters repair process, where the HRS completes remaining functions.

2.2 The hybrid repair strategy

A hybrid repair strategy (HRS) is presented here, which includes both the evolutionary repair method and the QMRT, for designing repair circuit. In the HRS, two processes are involved. The first one is the evolutionary repair process to repair many uncompleted functions with a few of gates. The next one is running the QMRT to repair remaining uncompleted functions.

The evolutionary repair method can repair many functions with a few of gates. But the stalling effect of evolutionary algorithm still exists when it is adopted to design repair circuit [6]. Therefore, it is not appropriate to continue adopting this repair method when the repair circuit is almost completed. The direct-repair method is adopted to generate the rest circuit in [6], while the QMRT is adopted in this paper.

The QMRT is also targeted at the circuit directly. And taking into account the characteristics of Quine-McCluskey algorithm, the number of circuit inputs has little effect on the repair time, so the repair time can be kept in an acceptable range. Meanwhile, the gate consumption is acceptable in a limited number of input-output combinations premise as well.

It is needed to be highlighted that evolutionary repair process is not always necessary in the HRS, as long as the fitness of current circuit reaches a certain standard, namely *Threshold_Fitness*, the QMRT works immediately. Relying on the QMRT, the HRS is always able to output repair circuit to meet the requirements, which means the success probability of HRS is 100%.

The pseudo code of HRS is shown in Fig. 3 and we make some brief explanations for the code as follows.

- 1) In the HRS, based on the fitness of *Uncompeled_Circuit*, it is the first process to determine that what kind of repair method is adopted to repair the uncompleted circuit. So, if the fitness of *Uncompeled_Circuit* is larger than *Threshold_Fitness*, the QMRT is adopted firstly in Step1, and the complete repair circuit is generated. If not, Step2 guides the algorithm to the evolutionary repair process.
- 2) From Step3, the HRS enters the evolutionary repair process. R_N records the evolutionary repair times. S_G records stalling times. Step4 initializes and runs EA. If the best fitness is larger than *Threshold_Fitness* in the evolutionary repair process, the QMRT will be adopted immediately in Step5, and the completed repair circuit is outputted. If not, the algorithm enters the stage that determining if the stalling effect occurs in Step8. Once the stalling effect occurs, the current evolutionary repair process needs to be stopped. So, Step9 and Step10 decide what kind of repair method is adopted in the next repair process. If the number of evolutionary repair is larger than *Max_Repair_Number*, the QMRT is triggered in Step9. If not, the evolutionary repair process continues in Step10.

HRS (<i>Uncompeled_Circuit</i> , <i>Stalling_Generation</i> , <i>Max_Repair_Number</i> , <i>Threshold_Fitness</i>)	
In:	<i>Uncompeled_Circuit</i> is the circuit to be repair. <i>Stalling_Generation</i> means the generations used to identify the stalling effect. <i>Max_Repair_Number</i> means the max number of generating the evolutionary repair circuits. <i>Threshold_Fitness</i> is the best fitness to trigger the QMRT process.
Out:	A complete repair circuit
1.	IF (the fitness of <i>Uncompeled_Circuit</i> is larger than <i>Threshold_Fitness</i>) Repair the current uncompleted circuit with the QMRT.
2.	ELSE
3.	$R_N=0$
4.	Randomly initialize the population with the parameter P (population quantity) and $S_G = 0$. Evaluate the population according to the truth table of repair circuit and evolve these individuals with EA.
5.	IF (the best fitness is larger than <i>Threshold_Fitness</i>) Repair the current uncompleted circuit with the QMRT.
6.	ELSE
7.	IF (the best fitness do not change) S_G++ .
8.	IF ($S_G \geq \textit{Stalling_Generation}$)
9.	IF ($R_N \geq \textit{Max_Repair_Number}-1$) Repair the current uncompleted circuit with the QMRT.
10.	ELSE Generate the truth table of the repair circuit according to the current incomplete circuit, R_N++ , and go to Step 4.

Fig. 3. The pseudo code of the HRS

3 An example

We will take an example to illustrate the proposed EA-HRS in this section.

In Table I, A , B , C and Out are input-output vectors; Out_e is the output of uncompleted circuit designed with EA; $Outer$ and $Outqr$ is respectively the output variable of evolutionary repair circuit and Quine_McCluskey repair circuit. Here, we set that the *Threshold_Fitness* is 60% of the maximal fitness value. According to the description of EA-HRS, there are three possibilities in the design process.

Table I. The input-output combinations of the related circuits

Input-Output Vectors				Possibility1		Possibility2			Possibility3		
<i>A</i>	<i>B</i>	<i>C</i>	<i>Out</i>	<i>Oute</i>	<i>Outqr</i>	<i>Oute</i>	<i>Outer</i>	<i>Outqr</i>	<i>Oute</i>	<i>Outer</i>	<i>Outqr</i>
0	0	0	1	0	1	0	0	1	0	0	1
0	0	1	0	1	1	1	0	1	1	0	1
0	1	0	1	1	0	1	0	0	0	1	0
0	1	1	1	1	0	1	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	1	0	0	1	0	0	1
1	1	0	0	0	0	1	1	0	0	0	0
1	1	1	0	0	0	0	0	0	1	0	1

- 1 Possibility1. If the fitness of uncompleted circuit is larger than *Threshold_Fitness*, the repair process will be triggered, where the evolutionary repair process is skipped and only the QMRT works. An example is given in Possibility1 column of Table I. Comparing *Oute* with *Out*, 62.5% (larger than 60%) functions of target circuit have been completed. So, it is next step that concluding the truth table of Quine-McCluskey repair circuit, i.e. *Outqr*, based on the repair principle [7], so,

$$Outqr = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + A\bar{B}C \quad (1)$$

As the key of the QMRT, Quine-McCluskey algorithm will find out all prime implicants and essential prime implicants to obtain the simplest expression of the repair function. $\bar{A}\bar{B}\bar{C}$ and $\bar{A}\bar{B}C$ are adjacent minterms, so are $\bar{A}\bar{B}C$ and $A\bar{B}C$. The prime implicant of the former group is $\bar{A}\bar{B}$ and the latter is $\bar{B}C$. So the simplest expression of Quine-McCluskey repair circuit is

$$Outqr = \bar{A}\bar{B} + \bar{B}C \quad (2)$$

The final Quine-McCluskey repair circuit is shown in Fig. 4(a).

- 2 Possibility2. If the stalling effect occurs during producing the uncompleted circuit, the repair process will be triggered, where the both repair methods work. In the evolutionary repair stage, as long as the fitness of repair circuit is larger than *Threshold_Fitness*, the evolutionary repair process stops, the QMRT works immediately. An example has been listed in Possibility2 column of Table I. *Outer* is the output of a best individual. The combination of uncompleted circuit and all evolutionary repair circuits has finished 62.5% of target functions. Namely, Input-Output Vectors, 0101, 0111, 1000, 1100 and 1110, have been completed. So, the QMRT is triggered, and the Quine-McCluskey repair process is similar to that in Possibility1.

$$Outqr = \bar{A}\bar{B} + \bar{B}C \quad (3)$$

The final Quine-McCluskey repair circuit is shown in Fig. 4(a).

- 3 Possibility3. The repair process starts taking the same prerequisite with Possibility2. In evolutionary repair process, the number of finished functions does not reach *Threshold_Fitness*. Thus, the evolutionary repair will not stop until the repair times reach *Max_Repair_Number*. The Possibility3 column of Table I shows an example. The combination of uncompleted circuit and all

evolutionary repair circuits only finished 50% of target functions. So, the Input-Output Vectors, 0001, 0010, 1011 and 1110, need to be repaired with the QMRT.

$$Outqr = \bar{A}\bar{B} + \bar{B}C + AC \quad (4)$$

The final Quine_McCluskey repair circuit is shown in Fig. 4(b).

Fig. 4. The repair circuit designed with the QMRT: (a) Possibility1 and Possibility2; (b) Possibility 3

The three possibilities are three operation modes of the EA-HRS. For different evolutionary result, different operation modes will be taken. The data in Table I is not fixed, but as an example to show the running process of proposed algorithm.

4 Experiments

The EA-HRS algorithm is tested by some circuits listed in Table II, and compared with the EHW algorithms proposed in [5], [6], [7], [4] and [10]. For fair comparison, all the EAs in these algorithms adopt GA and all algorithms are run in the same environment.

In these experiments, P (population quantity), P_c (crossover probability), P_m (mutation probability) and *Stalling_Generation* are respectively 150, 0.8, 0.015 and 50. The *Threshold_Fitness* is 90% of the maximal fitness value. The gates used here are NAND, XOR, OR, AND, NOR with 2 inputs and MUX. The MUX gate is only used to construct the multiplexer component. The all experiments were performed with a personal computer with a Dual Core (Pentium IV) at 2.60 GHz and 2 GB of RAM.

The parameters setting for each circuit are given in Table II. The Circuit_Layout means the size of chromosome in GA. The experiment results are presented in Fig. 5 and Table III, where “Gates” means the average gates consumption which has been converted to the amount of NAND-equivalent, “Time” stands for the average total time cost. Again, the all selected EAs are GA.

4.1 Experiment for small scale combinational logic circuits

From the histograms shown in Fig. 5, it can be observed that EA-HRS could design some circuits. What is more, compared to the algorithms proposed in [5] and [6], the EA-HRS has a better performance in time and gates consumption. Simultaneously, the EA-HRS spends fewer gates than the EHW algorithm proposed in [7].

Table II. The parameters setting for circuits

Circuit_Name	Circuit_Layout	Max_Repair_Number
Adder 3×3	1×200	1
Adder 4×4		
Adder 5×5	1×100	
Adder 6×6		
Adder 7×7		
Adder 8×8		
Mult 3×3	1×200	1
Mult 4×4		2
Mult 5×5		1
Mult 6×6	1×400	2
Mult 7×7	1×500	
Parity 13e	1×150	1
Parity 14e		2
Parity 15e		
Parity 16e	1×100	1
Parity 17e		
Parity 18e		
Parity 19e		
Parity 20e		

As for Parity 14e, the algorithm [7] takes 541.9 gates, 1932.264 s; the EA-HRS consumes 135.1 gates, 3227.934 s. Although the EA-HRS costs more time, the gates consumption of EA-HRS is much fewer than that of the algorithm [7].

As for Adder 4×4 , the EA-HRS needs 77.3 gates and 141.499 s. However, the algorithm proposed in [6] consumes 222.8 gates and 260.459 s. As for Parity 13e, the EA-HRS takes 76.8 gates, 1988.446 s, but the algorithm [6] takes 548.2 gates and 2433.363 s. Obviously, the EA-HRS takes fewer gates and less time than the algorithm presented in [6].

Compared to the algorithm proposed in [5], the EA-HRS has more advantages in time cost and gates consumption. For example, as for Mult 3×3 , the EA-HRS takes 143.8 gates and 94.925 s, but Gates of the algorithm [5] is 175.6 and the Time is 212.131 s.

4.2 Experiment for large scale combinational logic circuits

As shown in Fig. 5, the time consumption of Adder 4×4 , Mult 4×4 and Parity 15e become larger. Therefore, for designing larger scale combinational logic circuits, the EA-HRS is combined with GDD [4] and a compressing_outputs_technique [10] to reduce the time consumption. The design processes are divided four steps: firstly, the truth table of target circuit is rewritten with GDD [4] and the compressing_outputs_technique; secondly, the circuit corresponding to the new truth table is evolved by the EA-HRS; and then, the outputs of the evolved circuit are uncompressed [10]; finally, the outputs of original circuit are rebuilt with a multiplexer.

Fig. 5. The information of circuits designed with the three algorithms

In order to visually show the performance of EA-HRS in designing large scale combinational logic circuit, the experiment results are listed in Table III. In the experiments, adders and multipliers are divided into the size of 6 inputs. The parity check circuits are divided into the size of 12 inputs. Each experiment runs 10 times independently. The chromosome size of GA in the EHW algorithm proposed in [4] is 1×400 .

In Table III, the EHW algorithm [4] runs 3600 s and 12000 s for Mult 6×6 and Mult 7×7 respectively, but the both circuits are not completed successfully. From

Table III. The information of circuit designed with SA and GDD

	Gates			Time (s)		
	EA-HRS	[4]	[10]	EA-HRS	[4]	[10]
Adder 5×5	130.5	94.7	127.7	30.339	66.486	30.486
Adder 6×6	187.6	112.5	198.5	41.227	97.375	57.375
Adder 7×7	345.3	250.6	650.6	179.496	421.202	221.202
Adder 8×8	783.4	552.4	4095.4	971.934	2201.785	997.785
Mult 5×5	458.1	375	355.2	323.873	795.955	695.955
Mult 6×6	2956.6	—	2785.5	2831.167	>3600.000	3521.134
Mult 7×7	4998.3	—	4556.2	8412.627	>12000.000	10831.637
Parity 16e	60.7	33.3	63.3	32.521	60.590	30.590
Parity 17e	95.2	58.4	116.8	39.586	91.254	40.254
Parity 18e	132.6	75.8	157.5	62.824	184.625	84.625
Parity 19e	183.6	106.4	375.2	96.435	286.753	186.753
Parity 20e	246.5	161.3	526.3	125.154	423.211	323.211

Table III, although the EA-HRS takes more gates than the algorithm, it's obvious that the EA-HRS spends less time. For example, take Adder 6×6 , the EA-HRS costs 187.6 Gates and 41.227 s Time, but the EHW algorithm [4] takes 112.5 Gates and 97.375 s Time. As for Mult 5×5 , the EA-HRS spends 458.1 gates, but the algorithm [4] takes 375 gates only. However, in terms of Time, the EA-HRS needs 323.873 s, the algorithm costs 795.955 s.

Compared with the EHW algorithm proposed in [10], the EA-HRS shows a better performance in time consumption, and takes fewer gates in most instances. As for Adder 7×7 , the EA-HRS takes 345.3 gates and 179.496 s; the EHW algorithm [10] costs 650.6 gates and 221.202 s. As for Mult 6×6 , the SA consumes 2956.6 gates and 2831.167 s; the EHW algorithm takes 2785.5 gates and 3521.134 s.

5 Conclusions

In this paper, we present a novel hybrid repair strategy, i.e. HRS, which combines evolutionary repair method and QMRT and includes predefined value as one of flags to be triggered and change repair method. Based on the HRS, an evolutionary algorithm, i.e. EA-HRS, is put forward for the design of combinational logic circuits. In order to show the effectiveness of EA-HRS and SA, we implement some adders, multipliers and parity check circuits with related algorithms. Experimental results demonstrate that the EA-HRS can keep balance between the time cost and gates consumption very well, simultaneously, the EA-HRS has a better performance in gates and time consumption compared with the EHW algorithms proposed in [5] and [6], costs fewer gates than the EHW algorithm presented in [7], and spends less time obviously compared with the EHW algorithms outlined in [4] and [10].

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 61376025), the Industry-academic Joint Technological Innovations Fund Project of Jiangsu (No. BY2013003-11), the Funding of Jiangsu Innovation Program for Graduate Education (No. KYLX_0273), and the Fundamental Research Funds for the Central Universities.