An Online Evolvable Hardware System based on Hardware GA and PLA Structure

Yongkang Wei, Ning Wu, Xiaoqiang Zhang and Fang Zhou

***Abstract*—This work proposes a new evolvable hardware (EHW) system able to solve the problem of huge computation time involved in evolution of traditional EHW systems. In this paper, the speed of evolution of EHW has been obviously improved by shifting the implementation of genetic algorithm (GA) to hardware. Furthermore, the crossover and mutation operations of GA are optimized so that only one clock cycle is required to complete the operations. The virtual reconfigurable circuit (VRC) of programmable logic arrays (PLA) structure is adopted in the online evaluation of our system, which has the advantages of faster reconfiguration and less resource occupancy. In this paper, four kinds of actual circuit are used to test the proposed system. The results shows that compared to the traditional EHW systems, the speed of evolution and the speed of convergence are increased significantly.**

***Index Terms*—EHW system, PLA structure, GA, VRC**

I. Introduction

Evolvable hardware (EHW) is one kind of hardware with the ability to autonomously and dynamically change its own structure and behavior to adapt to the external environment. Due to its ability of adaptive, self-organizing and self-healing, there is an extremely wide range of applications in the fields such as circuit design, automatic control, fault-tolerant systems, pattern recognition and artificial intelligence, robot, deep space and deep-sea exploration etc. EHW consists of evolution module and evaluation module, the function of the former is to search for possible circuits to meet the requirements by genetic algorithm (GA), the function of the latter is to evaluate the explored circuit, and calculate the fitness of it. According to the different styles of evaluation, EHW can be divided into offline and online evolution. The fitness of individual is obtained by software simulation, and only the final result is downloaded to actual circuit to verify performance in offline evolution. However, each individual is downloaded to actual circuit to evaluate the fitness in online style. Because of the large amount of calculation and long time cost of software simulation, more importantly, there may be a large error between the simulation performance and the actual circuit, so this paper mainly studies the online EHW. At present, there are two main styles of implementation of EHW, one is PC+FPGA style [2,6], in which the evolution algorithm run in PC and evaluation is completed in FPGA, the other is CPU +FPGA style [3,7,8], in which the evolution algorithm run in embedded CPU core and the evaluation is completed in the same piece of FPGA.

Traditional EHW system faces the problem of huge computation time involved, which has been a major hindrance to real-time applications. In this paper, we shift the implementation of GA to hardware to speed up the evolution. We also optimize the crossover and mutation operations of GA, so that only one clock cycle is required to complete the operations. A strategy is proposed in GA to ensure that the best individual can always be transmitted to the next generation, which speeds up the convergence rate. A virtual reconfigurable circuit (VRC) with programmable logic array (PLA) structure is proposed in our system. And it has the advantages of faster reconfiguration and less resource utilization.

This paper is organized as follows: Section II gives a detail introduction to our EHW system. Section III shows some experimental results obtained by our system. Section IV concludes this work.

II. A New EHW System

A. *Framework of the EHW system*

Fig. 1 shows the framework of the EHW system. As shown in the figure, the EHW system consists of two parts, GA module and evaluation module. The function of the former is to search for possible circuits to meet the requirements by GA, the function of the latter is to evaluate the explored circuit and calculate the fitness of it.

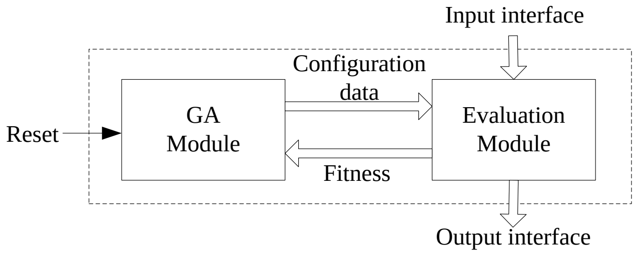

Fig. 1. Framework of the EHW system

There are two steps in the working process. Firstly, a new population is generated by GA module, and the individual, i.e.

Manuscript received June 10, 2015; revised July 31, 2015. This work was supported by the National Natural Science Foundation of China (No. 61376025, No. 61106018), the Industry-academic Joint Technological Innovations Fund Project of Jiangsu (BY2013003-11), the Funding of Jiangsu Innovation Program for Graduate Education (No. KYLX_0273).

Yongkang Wei is with the College of Electronic and information Engineering, Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing 210016, China (e-mail: 1072707966@qq.com).
Ning Wu is with the College of Electronic and Information Engineering, NUAA, Nanjing 210016, China (e-mail: wunee@nuaa.edu.cn).
Xiaoqiang Zhang is with the College of Electronic and Information Engineering, NUAA, Nanjing 210016, China (e-mail: zxq198111@qq.com).
Fang Zhou is with the College of Electronic and Information Engineering, NUAA, Nanjing 210016 China (e-mail: zfnuaa@nuaa.edu.cn).

the reconfiguration data, is sent to the evaluation module to configure the circuit. Secondly, the evaluation module evaluate the circuit and calculate the fitness, once the fitness is equal to the expected value, the system will configure the circuit and end the evolution, one can use the circuit through the input interface and the output interface.

B. *Hardware implementation of GA*

The structure of GA module is shown in Fig. 2. The module includes Initialization Module, Memory Module, Random Number Generator (RNG) Module, Selection Module, Crossover Module and Mutation Module. It is completely built on the configurable logic blocks (CLB) of field programmable gate array (FPGA).The coding is done using Verilog hardware description language (HDL). Xilinx ISE 14.4 has been used for synthesis and simulation. The Initialization Module is used to generate the initial population, and the original individuals in population are generated by RNG Module with the same probability of 0 and 1. In addition the RNG Module also needs to generate the operation data used in Crossover and Mutation modules with the probability of crossover and mutation respectively. The code and fitness of the individual is stored in Memory Module.

In this paper, selection operation, crossover operation and mutation operation have been optimized to speed up the evolution progress. The details of optimization are shown as follows:

- Selection Operation: Two individuals are randomly selected from the population every time, and then compare their fitness, only the better one will be sent to crossover operation.
- Crossover Operation: One operation data, generated by RNG module with crossover probability, is used to operate with individual every time, so that only one clock cycle is required to complete the crossover operation, and the speed is greatly improved. Here we have two individuals represented by A and B, and the operation data represented by O, A', B' represent the new individual generated through crossover operation. The crossover operation can be expressed by (1) and (2):

$$A' = (\sim O \& A) \,|\, (O \& B) \qquad (1)$$

$$B' = (\sim O \& B) \,|\, (O \& A) \qquad (2)$$

- Mutation Operation: One operation data, generated by RNG module with mutation probability, is used to operate with individual every time, so that only one clock cycle is required to complete the mutation operation, and the speed is greatly improved. The mutation operation can be expressed by (3):

$$A' = A \wedge O \qquad (3)$$

where A and O represent the individual and operation data, A' represent the new individual generated through mutation operation.

- In this paper, we proposed a strategy to ensure the best individual always be transmitted to the next generation by coping the best individuals in parent population and current population k and j times respectively, and it speeds up the convergence rate.

C. *Evaluation of the fitness*

Fast and efficient reconfigurable circuit is required for the real-time evaluation of online EHW. In order to complete the reconfiguration of circuit efficiently, a VRC of programmable PLA structure is adopted. The VRC which can overcome the limitations of different FPGA vendors, with the dynamic reconfiguration speed of nanosecond level, is built by Verilog-HDL.

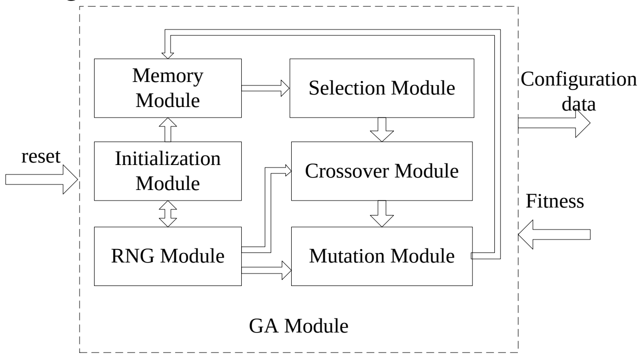

Fig. 2. Structure of GA Module

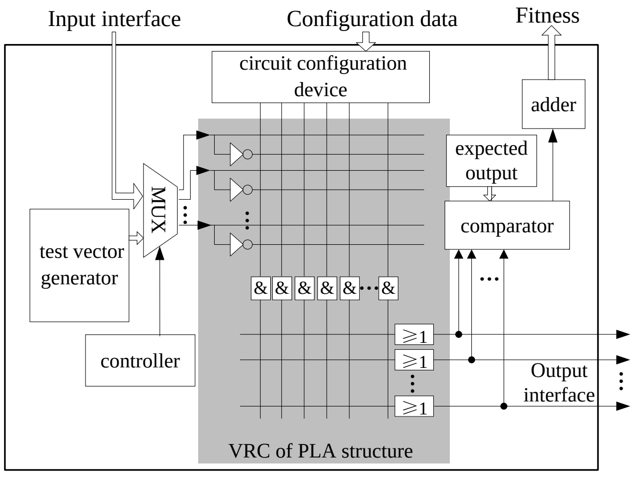

Fig. 3. Structure of Evaluation Module

The structure of evaluation module is shown in Fig. 3. It consists of VRC of PLA structure, test vector generator, controller, circuit configuration device, adder and comparator. PLA has a set of programmable AND gate planes, which link to a set of programmable OR gate planes. In this paper, we use '1' and '0' represent whether the intersections of horizontal wires and vertical wires are connected or not. All of the intersections are encoded as a binary string in the order of top to bottom and left to right. The encoding of individual is the binary string.

TABLE I
RESULTS OF SYNTHESIS

Resource	Slice	LUT	Reg.	Memory(KB)
Used	3203	9603	6855	414
Utilization	39%	29%	21%	8%

The process of evaluation as follows : Firstly, the individual's encoding is sent to circuit configuration device as the configuration data to reconfigure the VRC, then the test vector generator generates vectors used as the inputs of the VRC, after that, it calculates the fitness by comparing the output of VRC with expected output, last, sends the fitness to GA module. The controller controls the input of the VRC circuit, when the status is evaluation, the test vector generator is used as input, otherwise the input interface is used as the input. Taking a VRC of PLA structure with 4 inputs and 4 outputs for example, there are 16 test vectors and 16 combinations of output in total, so the expected fitness is 64.

III. Experiment Results

In this paper, we design a complete hardware system of EHW which speeds up the evolution obviously. The coding is done using Verilog-HDL. Xilinx Virtex5 FPGA is used to verify the feasibility and validity of the system. The frequency

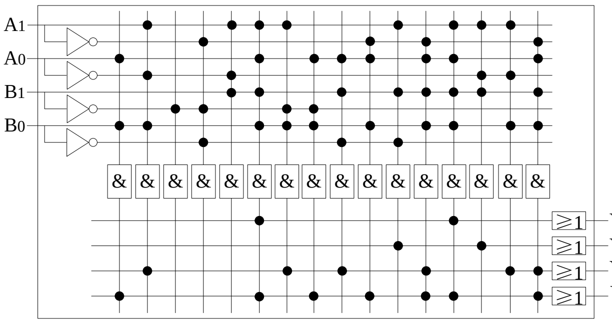

Fig. 4. The evolution result of multiplier with 4 inputs and 4 outputs

TABLE II
RESULTS COMPARED TO [2]

Inputs, outputs	function	paper	expected fitness	generations	time(s)	speed up
3 inputs, 3 outputs	adder	[2]	24	308	20	1069
		ours		100	0.0187	
4 inputs, 4 outputs	multiplier	[2]	64	16820	1200	8633
		ours		448	0.139	

TABLE III
RESULTS COMPARED TO [3]

function	paper	expected fitness	time for one generation(ms)	generation	speed up
2 bit adder	[3]	16	70	4507	903
	ours	48	0. 311	1123	
2 bit multiplier	[3]	16	70	95269	47864
	ours	64	0. 311	448	

represented by f is 33MHz, and the size of population represented by n is 500, and the probability of crossover and mutation represented by Pc and Pm respectively is 0.3 and 0.2. The number of best individual in parent population and current population represented by k and j respectively is 20 and 30. TABLE I shows the results of synthesis of the system.

In this paper, we chose a multiplier with 4 inputs and 4 outputs as the target circuits to evolve. The result of evolution circuit is shown in Fig. 4 .The relation expressions of inputs and outputs are shown in (4).

$$\begin{cases} Y_0=A_0B_0+A_1A_0B_1B_0+A_0\overline{B_1}B_0+\overline{A_1}A_0B_0+\overline{A_1}A_0B_1B_0=A_0B_0 \\ Y_1=A_1\overline{A_0}B_0+A_1\overline{B_1}B_0+A_0B_1\overline{B_0}+\overline{A_1}A_0B_1B_0 \\ Y_2=A_1B_1\overline{B_0}+A_1\overline{A_0}B_1 \\ Y_3=A_1A_0B_1B_0 \end{cases} \tag{4}$$

where A and B represent the operands of the multiplier. A_1 and A_0 represent the high and low bit of A. B_1 and B_0 represent the high and low bit of B. Y_0, Y_1, Y_2, Y_3 represent the result of A multiplied by B. We can verify that the (4) is the function of multiplier. It shows our system is effective and feasible. We also chose some commonly used examples such as adder with 3 inputs and 3outputs, adder with

TABLE IV
RESULTS COMPARED TO [7]

function	paper	generations	time for one generation(s)	time(s)	speed up
5 bit parity generator	[7]	198.5	39.8	7.9	14
	ours	998	0.559	0.558	

4 inputs and 3 outputs and parity generator with 5 bit. We run full test 20 times for each circuit. TABLE II, TABLE III, TABLE IV show the results compared to [2], [3] and [7].

Seen from TABLE II, to evolve the adder with 3 inputs and 3 outputs, platform in paper[2] needs 20 seconds while our system needs only 0.0187 second, the speed is improved by 1069 times. The speed of the multiplier with 4 inputs and 4 outputs is improved by 8633 times compared to [2]. In addition, the generations required in our system are less than them in [2].

Seen from TABLE III, the generations required in our system to evolve the 2 bit adder and 2 bit multiplier is 1123 and 448, it is less than 4507 and 95269 in paper [3]. It shows the effectiveness of the strategy proposed in this paper to speed up the convergence rate.

TABLE IV shows the evolution result of 5 bit the parity generator. It needs 39.8 seconds to complete one generation in [7] more than 0.559 second in our system, but the generations is 198.5 in [7] less than 998 in ours.

IV. Conclusion

A new EHW system is proposed to solve the problem of huge computation time involved. The speed of the evolution is improved greatly by shifting the implementation of GA to hardware. Furthermore we have proposed a new VRC of PLA structure which has advantages of less utilization resource and faster reconfiguration. We also adopt a strategy to speed up the convergence rate of GA. The system is completely implemented based on Xilinx Virtex5 FPGA. We have used the EHW system to evolve four kinds of circuit successfully, and it shows that our EHW system is much faster than the traditional ones in [2], [3] and [7].

References

[1] Garvie, Miguel, and A. Thompson, "Evolution of Combinatonial and Sequential On-Line Self-Diagnosing Hardware," *Evolvable Hardware, NASA/DoD Conference on* IEEE Computer Society, 2003:177.

[2] H. Yang, L. Chen, S. Liu, et al. "A Flexible Bit-Stream Level Evolvable Hardware Platform Based on FPGA," *Adaptive Hardware and Systems, 2009. AHS 2009. NASA/ESA Conference on* IEEE, 2009:51 - 56.

[3] Q. Zhang, J. Zhou, X. Yu, "An Evolvable Hardware Platform with the Ability of Local Dynamic Reconfiguration in Real-time," (In Chinese), *Journal of Northwestern Polytechnical University,* 29.5 (2011):761-765. DOI:10.3969/j.issn.1000-2758.2011.05.019.

[4] P Y. Chen, R D. Chen, et al. "Hardware Implementation for a Genetic Algorithm," *IEEE Transactions on Instrumentation & Measurement* 57.4(2008):699-705.

[5] P R. Fernando, S. Katkoori, et al. "Customizable FPGA IP Core Implementation of a General-Purpose Genetic Algorithm Engine,"

Evolutionary Computation IEEE Transactions on 14.1(2010):133 - 149.

[6] Silva B A, Dias M A, Silva J L, et al. “Genetic Algorithms and Artificial Neural Networks to Combinational Circuit Generation on Reconfigurable Hardware,” *Reconfigurable Computing and FPGAs, 2010 International Conference on. IEEE*, 2010: 179-184.

[7] F. Cancare, M. D. Santambrogio, and D. Sciuto, “A direct bitstream manipulation approach for Virtex4-based evolvable systems,” *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on IEEE*, 2010:853-856.

[8] A. Ren, W. Zhao, S. Tang, et al. “Implement of evolable hardware based improved genetic algorithm,” *Natural Computation (ICNC), 2011 Seventh International Conference on IEEE*, 2011:2112-2115.

[9] F. Cancare, M. Castagna, M. Renesto, and D. Sciuto. “A Highly Parallel FPGA-based Evolvable Hardware Architecture.” *In Proceedings of Int. Conf. on Parallel Computing*, in print, 2009.

[10] D. Levi, S. Guccione, “Geneticfpgas: Evolving stable circuits on mainstream fpgas,” *In Proceedings of the First NASMDOD Workshop on Evolvable Hardware* 1999.