The Development of Evolutionary Hardware System

Chih-Yung Chen [1+], Yueh-Fei Ho [2] and Jun-Juh Yan [1]

[1] Department of Computer and Communication, Shu-Te University, No. 59, Hengshan Rd., Yanchao Dist., Kaohsiung City 82445, Taiwan

[2] Department of Industrial and Business Management, Aletheia University, No.70-11, Beishiliao, Madou Dist., Tainan City 72147, Taiwan

Abstract. This paper presents a novel digital circuit evolutionary hardware system, which contains a microcontroller and a FPGA chip. In this scheme, the microcontroller can be the genetic algorithm kernel to evolve the digital circuit using the slicing structure and routing graph representation. In addition, the FPGA is applied to be the framework of reconfigurable random digital circuit matrix. The proposed system not only uses less code length for evolutionary hardware description, but also achieves the objective of intrinsic evolvable hardware. The proposed system could be taken as an alternative way for possible evolutionary hardware applications in the future.

Keywords: Evolutionary hardware, genetic algorithm, slicing structure and routing graph.

1. Introduction

Hardware can evolve automatically according to the different objective is a hot issue to be studied. Evolutionary hardware design (EHD) researches aim to develop the techniques which can be evolved automatically based on evolutionary algorithms. The concept of EHD was first initialized by John von Neumann [1] in 1950s. He submitted the idea that hardware should be able to have the function of self-repair and self-reproducing. Recently, the field programmable gate arrays (FPGAs) and complex programmable logic devices (CPLDs) have great progress to achieve the idea.

Hugo de Garis [2] proposed evolvable hardware (EHW) and Darwin machine. Subsequently, the EHW application researches immediately devoted by Higuchi et al. [3], Koza et al. [4], Miller [5] and Thompson [6]. The main idea of these studies is to integrate the evolutionary algorithms into the hardware encoding. The commonly algorithms used in these studies include genetic algorithm (GA), genetic programming (GP), evolutionary programming (EP), and evolution strategy (ES). These algorithms try to find the best hardware design based on the specific constraints and make the hardware structure can be adjusted automatically die to environmental changes.

Many logic-level hardware design methods e.g., Boolean algebra and Karnaugh map etc. have been used in solving the digital circuit minimization problem. However, those methods cannot effectively find the satisfactory solution when the problem dealt is big and complex. One of popular method in this field is called gate-level evolutionary hardware design (EHD). The goal of such research is to find the digital circuit for fitting required functionalities. Basically, the objective function is constructed by an n-input/m-output truth table, and then the table is integrated into an evolutionary structure (e.g., random circuit matrix) to evolve a combination logic circuit.

In this paper, a novel variable topology, including the new logic circuit representation and corresponding evolutionary algorithm, is proposed. Furthermore, a gate-level intrinsic evolution hardware system is taken as an illustration platform for the proposed approach to demonstrate its performance and validity.

2. Genetic Representation for Evolutionary Hardware

In most of evolutionary hardware studies, random circuit matrix is treated as the EHD framework of digital circuit design [7]. Generally, most logic circuit can be described by a two-dimensional logic cell matrix. The unit *Rij* of the matrix, where j is the logic gate stage and i is the unit number in each stage, could

+ Corresponding author. Tel.: +886-6158000 ext. 4807; Fax: +886-6158000 ext. 4899.
E-mail address: mikechen@stu.edu.tw.

be one of two-input type logic gates, such as AND, NOT, OR, XOR and WIRE. The inputs of each unit can be randomly connected by any element outputs in previous stages. Consequently, the 5×5 random circuit matrix and the evolutionary algorithm were used as the search frameworks to find the best solution. However, there are still some shortcomings and constraints such as the size setting of circuit matrix must rely on the designer's experiences.

2.1. Slicing Structure and Post-fix Expression for Digital Circuit Topology

Slicing tree is a kind of binary tree structure. In this structure, all terminated nodes of a tree are members of operands, denoted as $\pi = \{1,2,3,...,n\}$. All non-terminated nodes of a tree are members of operators denoted as $\theta = \{*,+\}$. It can transfer tree structure to slicing floor plan using infix traverse and usually starts from the left node. Operator '$*$' denotes the right node operand located on the right of left node operand as shown in Fig. 1(a). Operator '$+$' denotes the right node operand located on the top of left node operand as shown in Fig. 1 (b). This approach can transfer a complicated spatial relationship to become a binary tree structure as shown in Fig. 1 (c).

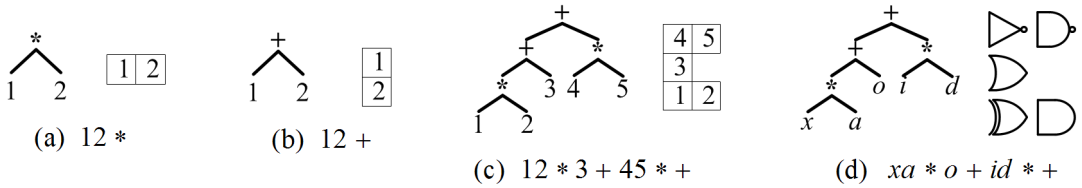

Fig. 1: Slicing tree and slicing floor plan.

Tree structure is usually stored by linking list as one-dimensional memory structure. Binary tree is an exception of tree structure. It can be shown by infix, prefix, or postfix expressions. The postfix expression was used in our study for evolutionary algorithm development due to its easy computational characteristic. For instance, the slicing tree shown in Fig. 1 (c) can be expressed as postfix string "$12*3+45*+$" and stored in the one-dimensional memory array. Then, such a string can be encoded as chromosomes and used in our evolutionary algorithm.

In the proposed approach, the slicing structure and postfix expression are used to develop the topology relationship of logic gates. All symbols of genetic encoding of logic elements are noted and listed in Table 1. Nine elements including input pin, output pin, and 7 logic gates were used in the proposed approach as shown in Table 1. For instance, the symbol "a" means an AND gate with two inputs and one output. Besides, symbols c and z represent input and output pins, respectively. These symbols can be used to show the relationship of terminated nodes in the slicing structure and also can change the space relationship of logic gates.

Table 1. Defined symbols for logic gates

Logic Gate	NOT	AND	OR	XOR	NOR	NAND	NXOR
Symbol	i	a	o	x	r	n	e

For example, the string "$xa*o+id*+$" is a legal postfix grammar description. The relationship of its slicing tree and mapping logic circuits is shown in Fig. 1(d). The result also proves that "$i*+aax*$" is an illegal postfix string.

2.2. Graph-based Routing Structures

In addition, the circuit routing is another problem needs to be solved. In this study, graph theory is used to approach this problem. Chen et al. [8] proposed a graph-based EHD technique called evolutionary graph generation (EGG) and applied it to the design of combinational and sequential arithmetic circuits. Also, the similar designing process is adopted in our studies.

Let $G=(V,E)$ denote a digraph (i.e., digital circuit). V is a nonempty set and $E\subseteq V\times V$. The pair (V,E) is then called the direct graph on V and E is the set of edges (i.e., wires). $V=V_1\bigcup V_2$ is the set of nodes (i.e., logic elements) and can be expressed as follows:

$$V_1=\{v_1,v_2,v_3,...v_n\} \text{ and } V_2=\{c_1,c_2,c_3,...c_m\} \quad (1)$$

where V_1 and V_2 are sets for non-input elements and input elements, respectively.

For representing a feedback free circuit, the routing graph G should be a loop-free directed graph. Thus, the edges of a logic circuit can be expressed as

$$E = \{v_i, x_i\} \bigcup \{v_i, y_i\} \tag{2}$$

where $1 \le i \le n$, $v_i \in V_1$, $x_i, y_i \in V$, and $v_i \neq x_i \neq y_i$.

Each gate of a circuit obtains its inputs from previous stages. Then, the constraints can be written as

$$\forall_{i \in I} \, \text{stage}(v_i) > [\text{stage}(x_i) \| \text{stage}(y_i)]. \tag{3}$$

Each logic cell should have two input pins, namely

$$\forall_{i \in I} \, \text{od}(v_i) = 2 \tag{4}$$

where $\text{od}(\cdot)$ means the outgoing degree.

Thus, the number of edges is

$$|E| = 2|V_1|. \tag{5}$$

Suppose a circuit routing graph $G' = (V', E')$ as shown in Fig. 2, which has nodes $V' = \{c_1, c_2, c_3, v_1, v_2, v_3, v_4\}$ and edges $E' = \{(v_1, c_1), (v_2, c_3), (v_3, v_2), (v_3, v_1), (v_4, v_3)\}$. In this graph, c_is are incoming nodes. Each outgoing node v_i only has maximum of two edges. The edges v_i must satisfy the constraint defined by equation (3). For example, the edge $\{v_1, v_3\}$ in graph G_1 is not a feasible edge due to $[\text{stage}(v_1) = 2] < [\text{stage}(v_3) = 3]$.

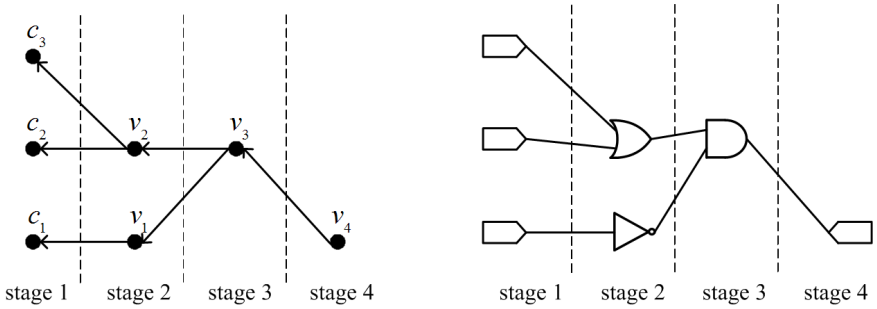

Fig. 2: Routing graph for digital circuit.

2.3. Genotype and Phenotype

A slicing genotype coding can be separated into three stages, i.e., input stage C, logic stage S_l, and output stage Z. Three stages then can be combined with two '$*$', and expressed as

$$S = CS_lZ\{**\}. \tag{6}$$

Finally, D is the genotype to represent a digital circuit with n-input, logic elements, m-output and routing graph G. A legal the mapping genotype code can be transformed into phenotype.

$$D = (S, G) \tag{7}$$

2.4. Evolution Operators

In this study, GA technique is used to execute the evolution process of EHD. The evolutionary operators of GA include initialization, mutation, crossover, fitness function and selection. The detailed description can be found in our previous study [9].

3. System Implementation

Refer to the study [10], a FPGA chip is designed to be digital circuit EHD framework using the same concept. Fig. 3 shows the proposed architecture of reconfigurable digital circuit. This scheme contains 20 evolvable logic elements. Each element can randomly connect to any output with two input pins.

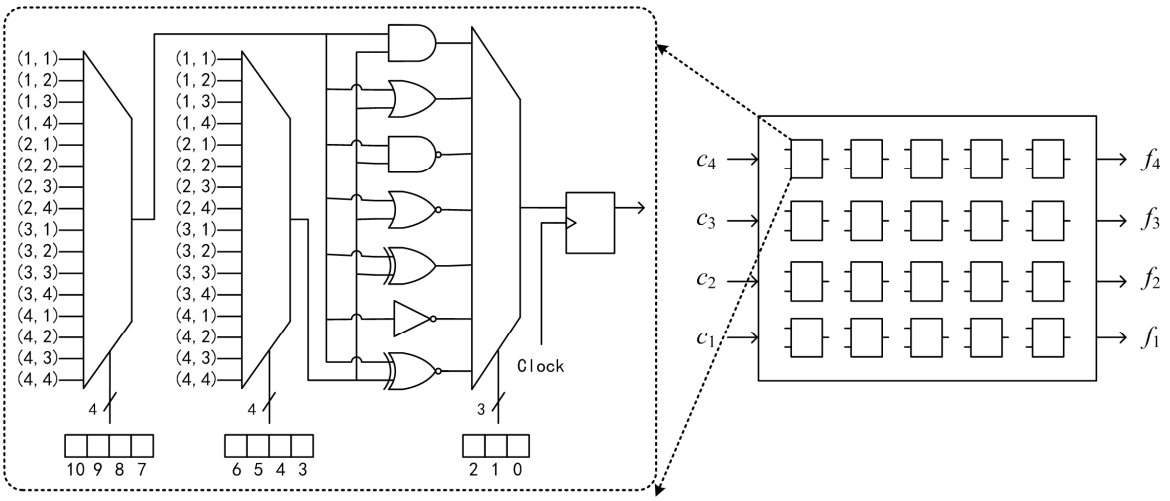

Fig. 3: Reconfigurable digital circuit.

The developed EHD system contains a microcontroller and a (FPGA) chip as shown in Fig. 4. A high-performance 8051 microcontroller is applied to genetic hardware kernel. All evolution processes are implemented in firmware to run the mentioned operators.

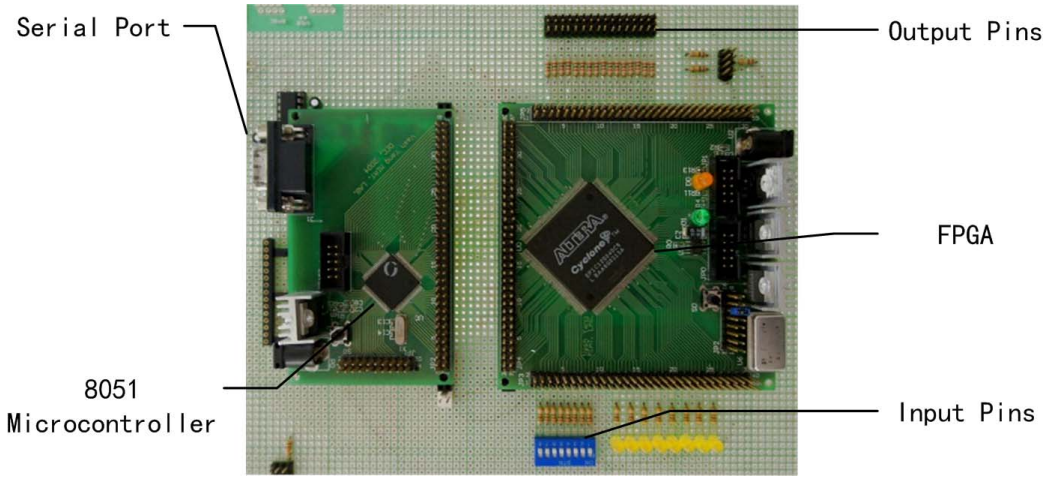

Fig. 4: The implementation of proposed evolutionary hardware system.

4. Experimental Results

To illustrate the details of the proposed approach, 14 examples of evolutionary hardware design were studied. Due to the page limitation, a 2×2 multiplier which has 4 inputs ($c_1c_2c_3c_4$) and 4 outputs ($f_1f_2f_3f_4$) is demonstrated in this paper. The representation of proposed postfix string which is found by the proposed evolutionary hardware system, is shown as following string,

$$c_1c_2c_3c_4 +++a_1a_2a_3 * x_1 * a_4 * a_5x_2 *++ f_1f_2f_3f_4 +++** . \tag{8}$$

And, the edges of routing graph are shown as follows.

$$\begin{aligned}&\{(a_1,c_2),(a_1,s_1)\},\{(a_2,c_1),(a_2,s_2)\},\{(a_5,c_2),(a_5,s_2)\},\{(a_3,a_1),(a_3,a_2)\},\\&\{(x_2,a_1),(x_2,a_2)\},\{(x_1,a_3),(x_1,s_1)\},\{(a_4,c_1),(a_4,x_1)\},\{(f_1,a_3)\},\\&\{(f_2,a_4)\},\{(f_3,x_2)\},\{(f_3,a_5)\}\end{aligned} \tag{9}$$

Practically, the best solution of 2×2 multiplier case was found with the population size of 50, crossover rate 0.7 and mutation rate 0.06 after 3500 iterations. The microcontroller is performed genetic operations within the working frequency 100MHz and 64K external memory. Fig. 5 shows the digital circuit diagram for this 2×2 multiplier approached by the proposed method.

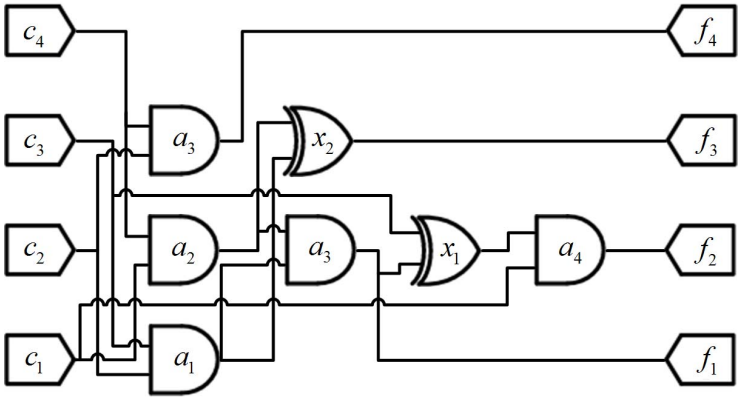

Fig. 5: The implementation of proposed evolutionary hardware system.

5. Conclusion

This paper proposes a new variable topology of EHD technique, which can be used for evolutionary computing, variable routing, and function block modification. Also, this study intends to design an intrinsic evolvable hardware to evolve adaptive digital hardware for a required truth table. Experimental results demonstrate the performance of the proposed scheme. For the future studies, the proposed system is expected to develop more intelligent systems such as self-reconfiguring, self-repairing, fault tolerant, design automation, intelligent robots, adaptable controller and reconfigurable hardware etc.

6. Acknowledgements

This work was supported by the National Science Council of Taiwan under contract No. NSC 100-2632-E-366-001-MY3.

7. References

[1] von Neumann, J. and Burks, A. W. *The theory of self–reproducing automata*, University of Illinois Press, Urbana. 1966.

[2] de Garis, H. Evolvable hardware: Genetic programming of a Darwin machine. *Artificial Neural Nets and Genetic Algorithms*, Springer-Verlag. 1993, pp. 441-449.

[3] Higuchi, T., Murakawa, M., Iwata, M., Kajitani, I., Liu, W. and Salami, M. Evolvable hardware at function level. *IEEE International Conference on Evolutionary Computation.* 1997, pp. 187-192.

[4] Koza, J. R., Bennett III, F. H., Andre, D., Keane, M. A. and Dunlap, F. Automated synthesis of analog electrical circuits by means of genetic programming. *IEEE Transaction on Evolution and Computation.* 1997, **1** (2): 109-128.

[5] Miller, J. F., Job, D. and Vassilev, V. K. Principles in the evolutionary design of digital circuits - part I. *Journal of Genetic Programming and Evolvable Machines.* 2000, **1** (1): 8-35.

[6] Thompson, A. An evolved circuit, intrinsic in silicon, entwined with physics. *First International Conference on Evolvable Systems, Lecture Notes in Computer Science*, Springer-Verlag. 1997, No. 1259.

[7] Louis, S. J. and Rawlins, G. J. E. Designer genetic algorithms: Genetic algorithms in structure design. *Fourth International Conference on Genetic Algorithms*, Morgan Kauffman, San Mateo. 1991, pp. 53-60.

[8] Chen, D., Aoki, T., Homma, N., Terasaki, T. and T. Higuchi. Graph-based evolutionary design of arithmetic circuits. *IEEE Transaction on Evolution and Computation*. 2002, **6** (1): 86-100.

[9] Chen, C. Y. and Hwang, R. C. A New Variable Topology for Evolutionary Hardware Design. *Expert Systems with Applications.* 2009, **36** (1): 634-642.

[10] Vasicek, Z. and Sekanina, L. An evolvable hardware system in Xilinx Virtex II Pro FPGA. *International Journal of Computer Application.* 2007, **1** (1): 63-73.