퍼지 및 지능시스템학회 논문지 2001, Vol. 11, No. 8, pp. 746-753

A Study on Embodiment of Evolving Cellular Automata Neural Systems using Evolvable Hardware

Kwee-Bo Sim and Chang-Bong Ban

School of Electrical and Electronic Engineering, Chung-Ang University

Abstract

In this paper, we review the basic concept of Evolvable Hardware first. And we examine genetic algorithm processor and hardware reconfiguration method and implementation. By considering complexity and performance of hardware at the same time, we design genetic algorithm processor using modularization and parallel processing method. And we design frame that has connection structure and logic block on FPGA, and embody reconfigurable hardware that do so that this frame may be reconstructed by RAM. Also we implemented ECANS that information processing system such as living creatures' brain using this hardware reconfiguration method. And we apply ECANS which is implemented using the concept of Evolvable Hardware to time-series prediction problem in order to verify the effectiveness.

Key Words : Evolvable Hardware, GA Processor, Cellular Automata, Neural Network, Time Series Prection

1. Introduction

General purpose processors provide a generic computational hardware for diverse applications. However, due to their architecture, they may show inferior computational performance. In contrast, application-specific integrated circuits (ASICs) exploit intrinsic characteristics of an application's algorithm that lead to a high performance. However, their algorithm mapping restricts the range of applications[1].

In contrast to general purpose processor where the structure is irreversibly fixed in the design process, Evolvable Hardware (EHW) is designed to adapt to changes in task requirements or changes in the environment, through its ability to reconfigure its own hardware structure dynamically and autonomously[2]. This capacity for adaptation, is achieved by employing efficient search algorithm known as genetic algorithm (GA). That is, we can say that evolvable hardware is union of genetic algorithm processor and hardware that reconfiguration is available.

Genetic algorithm based on evolutionary process of nature has been known as a method of solving large-scaled optimization problems with complex constraints in various applications[3]. Since a major drawback of the genetic algorithm is that it needs a long computation time, the hardware implementations of genetic algorithm processor (GAP) are focused on in recent studies. In Evolvable Hardware, genetic algorithm processor is essential because it requires the fast processing speed. We must consider complexity and performance of hardware at the same time to design genetic algorithm processor efficiently[4].

The main device used for Evolvable Hardware is the FPGA (Field Programmable Gate Arrays). Specially XC6216 is platform that has been used widely to students of this field since Adrian A. Thompson used in his experiment that was epoch-making breakthrough[5]. This device held the configuration in static RAM, meaning that it could be quickly accessed, read and changed. But since the configuration is made safe by restricting the connectivity of the logic block this limits the application that the device is suitable for complex arithmetic circuit. In addition, production of the XC6216 discontinued. In May 1998 Xilinx has announced the new Virtex family, which has many of the properties of the XC6216 but without some of the drawbacks. This device can be damaged if random configuration bit is downloaded in it. So it has been generally assumed that Evolvable Hardware was not possible on the Virtex devices unless the whole evolutionary algorithm included the Xilinx tools, foundation software[6].

In this paper, we design genetic algorithm processor and propose new method of hardware reconfiguration using Virtex device. Genetic algorithm processor is designed modularizing for efficient action and applied pipeline structure for parallel processing. The new method of hardware reconfiguration is that RAM and frame are implemented in device. RAM receives configuration bit from outside, and frame's inner structure changes by this value. In frame, there are

접수일자 : 2001년 10월 1일
완료일자 : 2001년 12월 1일
감사의 글 : 본 연구는 2001년도 서울시·중소기업청 산학연 공동기술개발 컨소시엄사업의 연구비와 과학기술부 뇌과학 프로젝트(Braintech21)에서 일부 연구비를 지원 받아 수행하였습니다. 연구비 지원에 감사 드립니다.

connection structure and logic block that are reconfigurable. And we apply these method and implemented Evolving Cellular Automata Neural Systems (ECANS)[7] that is information processing system such as living creatures' brain. Finally, we apply ECANS based on Evolvable Hardware to Exclusive-OR and Time-series Prediction problem to verify the effectiveness of the proposed scheme.

2. Evolvable Hardware

The hardware that can be reconfigurable, which is the semiconductor integrated circuit that user can change its structure as software, is available in many field because the semiconductor of various structure can be implemented without semi-conductor manufacturing process. Also, this shows possibility that can construct hardware system which is adaptive in the change of environment and robust against defects. Representative example of the hardware that can be reconfigurable is FPGA. FPGA can realize the performance of the hardware by downloading bit strings that decide the interior composition of any hardware. Recently, research of changing the structure of a hardware into adaptive one attracts public attention.

For the reconfiguration of a hardware, one of technologies that is getting attention by many people is to use genetic algorithms. The reasons that researchers prefer genetic algorithms are as following, genetic algorithms can use the configuration bit of a base architecture as chromosome, and genetic algorithms are simple and practically blind mechanisms of nature. In addition, genetic algorithms can be easily realized on hardware[8].

The hardware that the structure can be changed automatically by evolutionary algorithm is called an Evolvable Hardware[9]. We regard the bit string of the hardware structure as a chromosome of evolutionary algorithm and evolve the hardware structure on fitness. That is, FPGA has bit strings that decide the hardware structure and then, various logic circuit can be realized by converting the bit strings. Therefore, it can be applied to the method finding optimized solution through searching the converted bit strings. The basic concept of Evolvable Hardware is shown in figure 1.

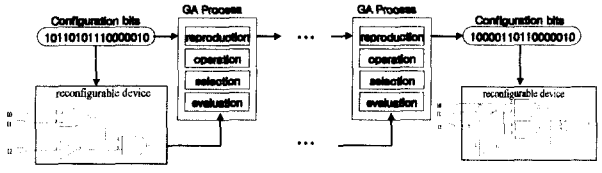

Fig. 1. Basic concept of Evolvable Hardware

Hardware design by evolution shows possibility for the system that existent circuit design methods couldn't or was difficult to design. When hardware design by evolution replaced existent circuit design, it has more advantages than existent circuit design which was handled by person. First, evolutional design can consider and search much more possibilities than existent design. Especially, it can be applied even though desired hardware was hard to be expressed. Second, evolutional design can be applied even if there is no particular base knowledge in problem. Third, it is comparatively easy that evolutional design techniques consider restriction condition and special requirement of various form, and it commonly is done through expression of chromosomes or fitness function.

3. Genetic Algorithm Processor

Evolution process of being of living things in natural world is that an individual that has high fitness for environment among the population, which is the set of individual that form certain generation, is survived by high probability. At this time, by crossover or mutation the population of the next generation is formed. Genetic algorithm imitates this evolution process of living things of natural world[3][10].

In genetic algorithm, population expresses candidate of solution about given problem. Genetic algorithm is creating population repeatedly by new population using crossover and mutation operator, and searches solution of problem. Individuals are selected according to their fitness and reproduce offsprings by crossover and mutation operator. At this time, individuals that have high fitness produce more offsprings averagely than individuals that have low fitness.

The genetic operations mimic the process of heredity of genes to create new offspring at each generation[3]. Major operators of genetic algorithm are crossover and mutation. The crossover operator creates new individual changing part of chromosome in selected individual. The mutation operator replaces allele with one or more genes of a selected chromosome. And this operator acts role that keeps variety of population.

Because genetic algorithm has simple structure and general method, it is of very wide application. But genetic algorithm needs much operational times because it passes through recursive evolution and adaptation process. Existent software base genetic algorithm had difficulty in applications requiring operational process of high speed. So we need hardware that is called genetic algorithm processor which has operational processing of high.

3.1 Structure of genetic algorithm processor

Efficient hardware embodiment method of genetic algorithm is required to reduce operational time that is

serious problem of genetic algorithm[4]. And it must equip structure that consider complexity and performance of hardware at the same time.

In this paper, we design genetic algorithm processor using modularization in this point. And we design it using series processing in a generation. Block diagram of proposed genetic algorithm processor is shown in figure 2. In proposed processor, there are operating module which do crossover and mutation function that is the basic operator of genetic algorithm, re-production module for reproduction of population, two memories which store population, and one memory that store fitness and genetic algorithm controller.

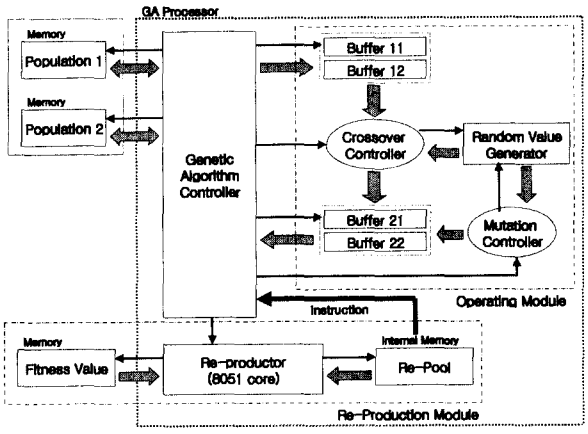

Fig. 2. Block diagram of genetic algorithm process

Proposed genetic algorithm processor produces a command to run processor in reproduction module and stores it in regeneration room (Re-Pool), thereafter, controller performs 'operation between individuals', 'read memory', and 'write memory' using the produced instructions. Also, genetic algorithm processor is designed to perform various genetic algorithm using 8051-core in reproduction module. That is, if user programs instruction production method which is made on reproduction process and downloads the codes into 8051-core, diverse genetic algorithm will be run on those instructions. And, by using two memory as storage for individuals, it is designed to perform both steady-state model and generation model genetic algorithm. In the case of generation model, it stores population for each generation with memory position exchanged, and for steady-state model, it uses only one memory. Above process is decided by instructions. Fitness values that corresponds to each individuals are stored in the third outer memory. That reason is to let the processor refer only fitness value when it reproduce individual to compose new generation.

3.2 Re-production module

Re-production module stores individuals which have to be reproduced in Re-Pool by referring the fitness values that are stored in memory.

At this time, the module stores one instruction composed of the ID of the two parent individuals for crossover, Op-Code, and Direction-Code which assign the storing position of the offsprings made by crossover and mutation. Figure 3 shows this instruction format and the ID is the storing position value in memory. Op-code controls a method of crossover and mutation operation and an transfer method of individuals. And ID and direction code display a relative position value which is stored or will be stored within memory. If it is stored individuals that are reproduced in the next generation are stored by this instruction format, the controller loads and executes the instructions. After it reads two individuals in memory referring two individual IDs, the crossover and mutation of two individuals are executed by referring the op-code. And, the result of the execution is stored orderly in the other memory referring direction code (Direction 1, 2).

Op-Code	Parent 1 ID	Parent 2 ID	Direction1	Direction2

Fig. 3. Instruction format

The created instructions decide the method of crossover and mutation operation that is the operations of individuals, and the transfer method of individuals. Therefore, genetic algorithm processor can do various genetic algorithms according to the created instructions. User must download the program that produce the instructions which execute the genetic algorithm that he wants to run at 8051 core in re-production module.

For example, in case of generation model, user stores it by instruction form in reproducing space after reproducing all individuals which have to be reproduced in next generation, executes these all instructions in the controller, and reproduce a instruction in reproducing module again.

3.3 Operating module

Operating module does crossover and mutation which is basic operation of genetic algorithm. And, because operating module does mutate right before crossover of parent individuals, it c an have the fast processing speed. Crossover operation does simply crossover and two point crossover. Figure 4 shows internal structure of operating module.

First, it reads two parent individuals that is target of crossover operation from memory and stores them in Buffer11 and Buffer12. At the same time, in crossover point generator, random number between 0 and 1 is generated. And if this number is small more than p_c (the probability of crossover), crossover point is generated, and crossover point is set 0 otherwise. At this time, one crossover point is generated in case of

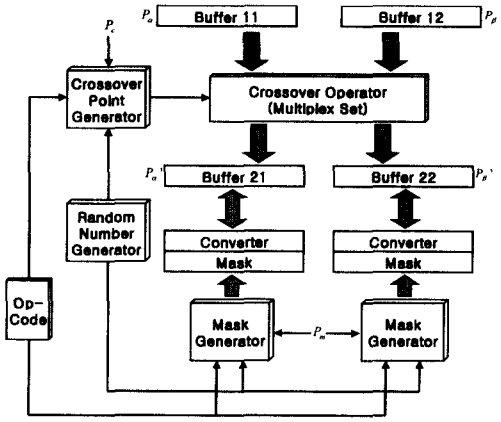

Fig. 4. Internal structure of operating module

simple crossover, or two crossover point are generated in case of two point crossover by referring op-code. Crossover point is generated randomly by integer between 1 and the length of chromosome-1. Crossover operator receives two parent individuals and crossover point and does crossover operation. And the offsprings after operation are stored in Buffer21 and Buffer22.

Mutation operation does point mutation. When the offsprings after crossover operation is stored, mask generator generates the random number between 0 and 1. And mask bit is set by 1 if the number is smaller than p_m (the probability of mutation), and it is set by 0 otherwise. But all mask bits are set by 0 if op-code is the mode that lets do not mutation. At this time, we set mask bits by the length of chromosome and create mask. Converter converts the bit if mask bit is 1, and does not otherwise. Offsprings after mutation operation are stored in Buffer21, Buffer22.

4. Design of ECANS using EHW

4.1 Hardware reconfiguration method

Evolvable Hardware is a combination of genetic algorithm processor and reconfigurable hardware; it is a evolutionary system which regards configuration bit deciding inner architecture as individual of genetic algorithm. The representative example of reconfigurable hardware is FPGA (Field Programmable Gate Arrays). It is a kind of Integrated Circuit (IC) consisting of many uncommitted programmable logic gates and uncommitted programmable connection line. And it can reconfigure its gates and connection line for conducting the function of end-user's need. Especially XC6216 is platform that has been used widely to researchers of this field since Adrian A. Tompson uses it in his experiment that was epoch-making breakthrough[11]. This device held the configuration bit in static RAM, meaning that it could be quickly accessed, read and changed. And the advantage of XC6216 is that researchers are able to convert device's configuration simultaneously without synthesis tool, because it has a sufficient capacity and its configuration bit is opened for reconfiguration in differentiating existing FPGA. In addition, it has a merit that user can reconfigure partly on device's working.

But since the configuration is made safe by restricting the connectivity of the logic block, it limits the application that the device is suitable for complex arithmetic circuit, In addition, its production is discontinued. So, we need a way of reconfiguration using Virtex. Xilinx has announced the new Virtex family, which has many of the properties of the XC6216 but without some of the drawbacks[6]. In the case of Virtex, we are hard to use existing method because of not opening configuration bit consisting of the inside. If user know its configuration bit, users cannot convert the configuration bit intentionally because Virtex is not a device for Evolvable Hardware but a device using harmless bit made by synthesis tool.

In this paper, for developing Evolvable Hardware we design a frame which user want to make in Virtex. The frame has connection structure and logic blocks. And we use a way of configuring status of the frame by using RAM which is constructed by user or internal RAM of device.

This way is widely used by developing a frame fitting matters of given matters of given problem, and downloading configuration bit form external to RAM inside of device. It is possible to evolve by evolutionary algorithm because every users who design the frame know configuration bit and reconfigure a inner structure partly because only some section of RAM can download a configuration bit again. In the connection structure in frame, we can configure the status of it by tri-state buffer. When a configuration bit is downloaded to RAM, tri-state buffer should work by following its value. In logic block, we use Look-Up Table (LUT). After a configuration bit is downloaded to RAM, it prints the value of bit by its value. Figure 5 shows a connection structure and design of logic block using RAM.

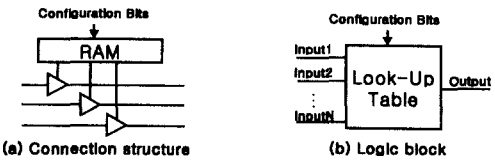

Fig. 5. Structure in frame

4.2 Implementation of ECANS

ECANS[7] is a neural net model based on evolution and development as information processing system such as living creature's brain. In this paper, we embodied ECANS by the concept of Evolvable Hardware. We use genetic algorithm processor to run genetic algorithm,

and each cell of cellular automata neural system is designed using hardware reconfiguration method proposed in this paper.

Evolving cellular automata neural system is based on development and evolution, in other words, it is modeled on the ontogeny and phylogeny of natural living things. A network is developed by cellular automata rule[12], and evaluated in given environment. And genetic algorithm takes a part in adaptation process.

A neuron of ECANS is used Nagumo-Sato's chaotic neuron model[13] that is based on Hodgkin-Huxely equation which is deep consideration of practical neuron membrane's characteristics. Modified equation of Nagumo-Sato's chaotic neuron model are as follows.

$$\left.\begin{aligned} y(t+1) &= u(x(t+1)) \\ x(t+1) &= I(t) - \alpha \sum_{d=0}^{t} k^d y(t-d) - \theta \end{aligned}\right\} \quad (1)$$

where, $y(t)$ is output at time t, $x(t)$ is inertia state at time t, $I(t)$ is input at time t, $u(\cdot)$ is unit step function, k^d is damping factor of refractoriness having values between 0 and 1, the constant α is positive parameter, and θ is threshold of neuron.

One neuron of networks has chaotic dynamics and some connection of neuron also shows chaotic behavior. Because action function of neuron shows in form of unit step function in chaotic neuron model, input/output signal is pulse form. Therefore, century of signal is measured by density change of pulse.

ECANS uses cellular automata rule for the development. This rule becomes chromosome through fixed DNA coding method, and becomes individual of evolution algorithm. If cellular automata rule is decided, network is developed according to cellular automata rule from initial cells. Block diagram of ECANS is shown in figure 6. Figure 7 shows developmental level of ECANS.

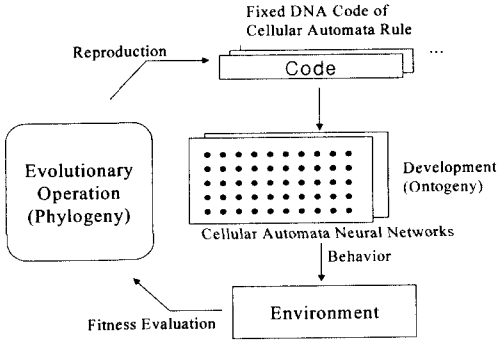

Fig. 6. Block Diagram of ECANS

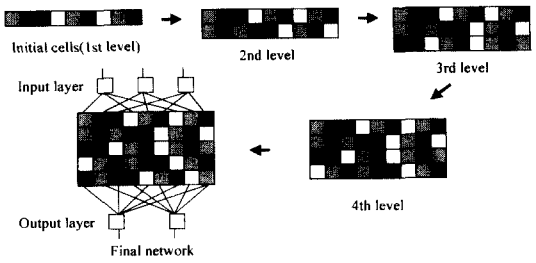

Fig. 7. Developmental level of ECANS

In this paper, we implement ECANS using the concept of Evolvable Hardware. Block diagram of the system is shown in figure 8. Genetic algorithm processor runs genetic algorithm. Rule generator analyzes each individual and produces cellular automata rule and inner structure of cellular automata neural network alters by this rule naturally. If cellular automata rule is generated in rule generator, it is stored in rule table of cellular automata neural network. And we design the cells in cellular automata neural network so that may compose automatically connection between the cells according to this rule.

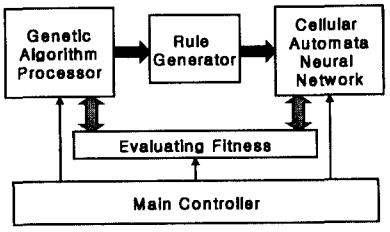

Fig. 8. Block diagram of System

We implement cellular automata neural network using FPGA. Figure 9 shows the structure of one cell of cellular automata neural network. Each cell of cellular automata neural network is consisted of cellular automata module, connection module, and neuron module. Cellular automata module and connection module are designed by proposed hardware reconfiguration method. The rule which is output of rule generator is stored to rule table of cellular automata neural network, and connection state of connection module is decided by this value. Figure 9 shows proposed structure of cell in cellular automata neural network.

The number of input of neuron is four. They are input from upper neuron, input from same layer's left and right neurons, and input from own neuron. And the right term of equation (1) was divided into equation (2).

$$\left.\begin{aligned} a_i(t+1) &= k_e a_i(t) + \sum_{j=0}^{M} v_{ij} I_j(t) \\ b_i(t+1) &= k_f b_i(t) + \sum_{j=0}^{N} w_{ij} y_j(t) \\ c_i(t+1) &= k_r c_i(t) - \alpha y_i(t) - \theta_i(1-k_r) \end{aligned}\right\} \quad (2)$$

where w_{ij} is weight value between j-th neuron and i-th neuron in current layer. v_{ij} is weight value between i-th neuron in current layer and j-th neuron in upper layer. k_e, k_f, k_r are real values between 0 and 1. And each of these value is represented the damping factor of external input term, feedback input term, and self-feedback term, respectively. θ_i is threshold of i-th neuron.

In neuron module, these three values are embodied by A, B, C Register and last state value is stored State Register. With a clock phase, each register shifts its own value for multiplication and adds this value to its input value. In cellular automata module, the state of cell is decided and control signal is sent to connection module. In connection module, whether the output and inputs of the neuron are connecter or not is determined by considering the state of cell decided by cellular automata rule.

That is, genetic algorithm processor regards coded cellular automata rule as individual and runs genetic algorithm. And cellular automata neural network refers cellular automata rule and changes inner structure automatically. Therefore, we design Evolvable Hardware that is combination of genetic algorithm processor and cellular automata neural network.

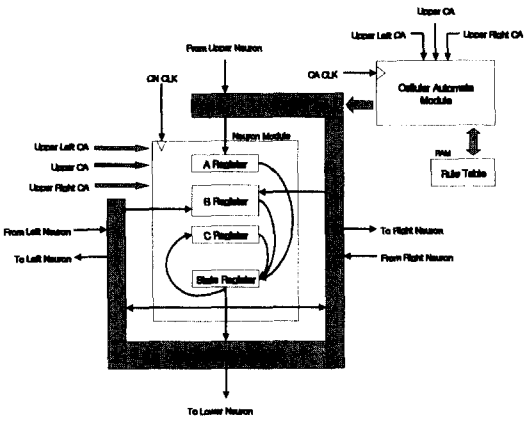

Fig. 9. Internal Structure of Cell

5. Experiment Results

Time-Series Prediction is that forecast future value with data that is obtained at past. The method of Time-Series Prediction is various with neural network, genetic programming, etc. If we do that given past data is x, result that is obtained by predictor is $\tilde{x}$, past set of data is vector x, we can express as following.

$$x(t)=(x(t),x(t-1),\cdots,x(t-\phi)) \tag{3}$$

Using past data as much as ϕ, $\tilde{x}(x(t))$, value that predict $x(t+1)$, can be obtained. That is, future value, $x(t+\tau)$ can be gotten by value $\tilde{x}(x(t))$ that predicted using past value. If $\tau=1$, we say as shot-term prediction, and we say as long-term prediction if $\tau\geq 2$.

In this paper, we applied implemented ECANS based on evolvable hardware to time-series prediction problems. As experiment data, we use Mackey-Glass chaotic data. Mackey-Glass' bloodstream equation[14] is representative chaos system. And this equation is used by standard equation that compares the performance of time series prediction mainly. Mackey-Glass equation is given in form of differential equation such as equation 6.

$$\frac{dx(t)}{dt}=\frac{ax(t-\Delta)}{1+x^c(t-\Delta)}-bx(t) \tag{4}$$

where a, b and c are typically fixed at 0.2 0.1 and 10 respectively. The Δ is a special parameter that we can alter how pathological the resulting time series is. For now, we use a τ of 30. These set contain 1000 point data. The half of it is used for training data and the rest of it is test data. Table 1 shows parameters that used in this experiment. Figure 10 is one of the result predicted by our proposed neural network and figure 11 is the structure of the neural network that is obtained at this time. MSE (Mean Square Error) of this is 0.0052, and fitness value is 0.9677.

Table 1. The parameters used in TSP

Parameter	Value
embedding dimension	5
time delay	5
network size	5 × 10
inputs	$y(t)$, $y(t-5)$, $y(t-10)$, $y(t-15)$, $y(t-20)$
output	$y(t+1)$
fitness function	$fit=10^{-10E}$, $E=\frac{1}{N}\sum_{t=0}^{n}(d(t)-y(t))^2$

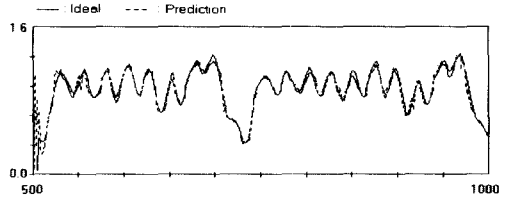

Fig. 10. Predicted and ideal Mackey-Glass data

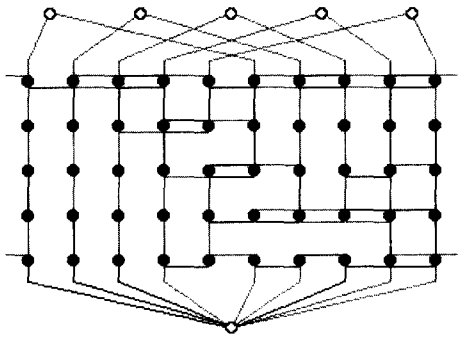

Fig. 11. Obtained evolutionary neural network for Time-Series Prediction
(Each black circle represents figure 9)

6. Conclusions

In this paper, we reviewed the basic concept of Evolvable Hardware that is combination of genetic algorithm and reconfigurable hardware.

And we considered complexity and performance of hardware to design genetic algorithm processor. This processor has three modules and memories. And we proposed hardware reconfiguration method. We designed the frame which has connection structure and logic block on FPGA. This frame is reconstructed by downloading configuration bit to memory. Finally we implemented ECANS that is information processing system such as living creatures' brain using proposed method. And we applied it to time-series problem and show its effectiveness.

In the future we will apply it to many application, and also the genetic algorithm processor and the method of hardware reconfiguration should be studied more generally.

References

[1] A. Stoica, D. Keymeulen, C. Lazaro, W. Li, K. Hayworth, and R. Tawel, "Toward On-Obard Synthesis and Adaptation of Electronic Functions: An Evolvable Hardware Approach," Proc. of 1999 Aerospace Conference, pp. 351-357, 1999.

[2] X. Yao and T.Higuchi, "Promises and challenges of evolvable hardware," IEEE Transactions on System, Man & Cybernetics Part C: Applications & reviews 1997., Vol. 29, No. 1, pp. 87-97, 1999.

[3] J.H. Holland, Adaptation in Natural and Artificial Systems, The Unviersity of Michigan Press, 1975.

[4] S. D. Scoot, A. Samal, and S. Seth, "HGA : A haredware-based genetic algorithm," Proc. ACM/SIMDA 3rd International Symposium on FPGA, pp. 53-59, 1995.

[5] A. Thomson, "An evolved circuit, intrinsic in silicon, entwined with physics," Procedures of the 1st international conference on Evolvable systems, Springer, 1996.

[6] G. Hollingworth, S. Smith and A. Tyrrell, "Safe Intrinsic Evolution of Virtex Devices," Proc. of the Second NASA/DoD workshop on Evolvable Hardware, pp. 195-202, 2000.

[7] D.W. Lee, K.B. Sim, "An Evolution of cellular automata neural systems using DNA coding method," Journal of IEEK, vol. 36-S, no. 12, pp. 10-19, 1999.

[8] M. Perkowski, A. Chebotarev, and A. Mishchenko, "Evolvable Hardware or Learning Hardware? Induction of State Machines from Temporal Logic Constraints," Proc. of the First NASA/DoD Workshop on Evolvable Hardware, pp. 129-138, 1999.

[9] X. Yao, et al., "Promises and Challenges of Evolvable Hardware," *IEEE Tran. on System*, Man and Cybernetics-Part C: Applications and Reviews, vol. 29, no. 1, pp. 87-97, 1999.

[10] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, 1989.

[11] A. Thompson, "On the Automatic Design of Robust Electronics Through Artificial Evolution," Proc. 2nd International Conference on Evolvable Systems: From biology to hardware, Springer-Verlag, pp. 13-24, 1998.

[12] C.G. Langton, "Life at the Edge of Chaos," Artificial Life II, Addison-Wesley, pp. 41-91, 1992.

[13] 合原一幸 編著, ニュラルシスツムにおけるカオス, 東京電氣大學出版局, 1993.

[14] M.C. mackey, L. Glass, "Oscillation and Chaos in Physiological Control Systems," Science 1977, p. 287, 1977.

저 자 소 개

심귀보(Kwee-Bo Sim)

1984년 : 중앙대학교 전자공학과 공학사
1986년 : 동 대학원 전자공학과 공학석사
1990년 : The University of Tokyo 전자공학과 공학박사
1997년~현재 한국퍼지 및 지능시스템학회 편집이사 및 논문지 편집위원장
1997년~현재 한국퍼지및지능시스템학회 편집이사
2000년~현재 제어자동화시스템공학회 논문지 편집위원 및 직선평위원
2000년~현재 대한전기학회 제어 및 시스템 부문회 편집위원 및 학술이사
1991년~현재 중앙대학교 전자전기공학부 교수

관심분야 : 인공생명, 진화연산, 지능로봇시스템, 뉴로-퍼지 및 소프트 컴퓨팅, 자율분산시스템, 로봇비전, 진화하드웨어, 인공면역계 등
Phone : +82-2-820-5319
Fax : +82-2-817-0553
E-mail : kbsim@cau.ac.kr

반창봉(Chang-Bong Ban)

2000년 : 중앙대학교 전자전기공학부 공학사
2000년~현재 동 대학원 전자전기공학부 석사과정
관심분야 : 진화하드웨어, 지능로봇 등