한국 퍼지 및 지능시스템학회 논문지 2000, Vol. 10, No. 4 (2000)379~389

패턴 인식을 위한 진화 셀룰라 분류기

Evolvable Cellular Classifiers for Pattern Recognition

주재호 · 신윤철 · 강 훈

Jae-Ho Ju, Yoon-Cheol Shin and Hoon Kang

School of Electrical & Electronics Engineering, Chung-Ang University

ABSTRACT

A *cellular automaton* is well-known for self-organizing and dynamic behaviors in the field of artificial life. This paper addresses a new neuronic architecture called an *evolvable cellular classifier* which evolves with the genetic rules (chromosomes) in the non-uniform cellular automata. An evolvable cellular classifier is primarily based on *cellular programming*, but its mechanism is simpler because it utilizes only mutations for the main genetic operators and resembles the Hopfield network. Therefore, the desirable bit-patterns could be obtained through evolutionary processes for just one individual agent. As a result, an *evolvable hardware* is derived which is applicable to classification of bit-string information.

1. Introduction

Cellular automata are a class of discrete spatio-temporal devices in which the spatial states and/or the genes of each cell evolve with time. The behaviors of cellular automata are so complex that even chaotic dynamics could occur, and if conditions are satisfied, emergent behaviors might be shown [1]. Historically, one can find the origin of cellular automata from von Neumann's self-organizing machines[2]. Conway's famous game of life[3] is also a by-product of cellular automata which reveals the fact that local interactions may end up with global life-like behaviors. Codd's cellular machines[4] influenced deGaris' evolving cellular brain machines[5]. Wolfram analyzed cellular automata and classified those into 4 categories[6], and Langton's loop[7] is another example that shows self-organization. Mitchell, Crutchfield, and Harber have investigated the dynamic behaviors of evolving cellular automata[8]. Sipper has solved several problems such as density, synchronization, and ordering by using his own co-evolved parallel cellular machines in terms of *cellular programming*[9].

This paper is concerned with cellular programming but is rather modified based on *Hopfield networks*[10] from the viewpoint of cellular automata. The main objective is to propose a new paradigm of evolvable neuronic architecture based on cellular automata which classifies binary information and solves genetic rules given by the prescribed bit-patterns of the cell states. Therefore, given any distributions of initial states, the genetic rules that have the cell states converge to the exact bit-string, are evolved through mutations in non-uniform genetic rules with an evolvable cellular classifier. The proposed paradigm differs from *cellular programming* because it doesn't use any crossover operators nor co-evolution processes.

2. Synthesis of Evolvable Cellular Classifiers

2.1 Non_uniform Cellular Automata

First, consider the design procedure of an evolvable cellular classifier. Each cell state consists of 1-bit data in {0, 1} and usually, the number of the neighborhood is 3 in 1-D case and 5 (center, north, east, west, south) in 2-D case of the von Neumann neighborhood. The dynamical behavior of each cell is governed by an implicit rule set called a "gene", 8-bits in 1-D case and 32-bits in 2-D case, respectively, distributed in a non-uniform lattice. Figure 1 shows how a cell state is computed logically by using the neighborhood

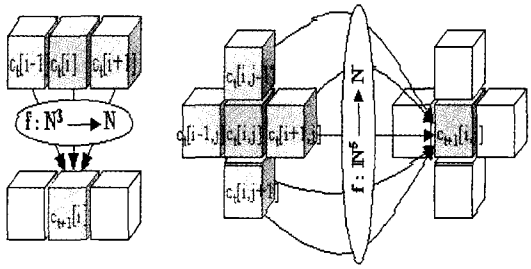


Fig. 1. Local Function in Cellular Automata

surrounding itself. The boundaries of cellular automata are cyclic in 1-D case and toroidal in 2-D case.

2.2 Structure of Evolving Cellular Classifiers

The cell states in evolvable cellular classifiers are represented by

$$c_{t+1}[i] = f_{1D}(c_t[i-r], \cdots, c_t[i], \cdots, c_t[i+r]) \quad (1a)$$

$$c_{t+1}[i,j] = \begin{bmatrix} & c_t[i,j-1] & \\ c_t[i-(1,j)] & c_t[i,j] & c_t[i+1,j] \\ & c_t[i,j+1] & \end{bmatrix} \quad (1b)$$

where $c_t[i]$ is the i-th cell state at time t, and f_{1D}, f_{2D} are functions of 1-D cellular automata with $(2r+1)$ neighborhood, 2-D cellular automata with von Neumann neighborhood, respectively. The functional operations are performed in parallel at each time t. Each cell in cellular automata contains the genotypes of an 8-bit binary genetic rule in 1-D case, and a 32-bit binary gene in 2-D case, which are described by

$$\text{gene}_{t+1}[i] = g(\text{local\_fitness}_t[i], \text{gene}_t[i], \langle\text{other\_genes}\rangle) \quad (2a)$$

$$\text{gene}_{t+1}[i,j] = g(\text{local\_fitness}_t[i,j], \text{gene}_t[i,j], \langle\text{other\_genes}\rangle) \quad (2b)$$

where g is the genetic operator, gene_t is a genetic rule at time t, and local_fitness_t is a 1-bit local fitness value in $\{0, 1\}$ at time t. An optional input argument <other genes> is taken into account when co-evolved evolutionary processes are involved. The local fitness is obtained by the exclusive NOR (XNOR) operation given by

$$\text{local\_fitness}_t[.] = \text{XNOR}(c_t[.], \text{desired\_bit}[.]) \quad (3)$$

where desired_bit[.] is an arbitrarily chosen desirable bit-pattern in {0, 1} related to the real-world phenotypical representations.

2.3 Evolutionary Process in Evolvable Cellular Classifiers

In brief, the block diagram of an evolvable cellular classifier is shown in Figure 2 in which the dynamic behaviors of cell states are controlled by neighboring cell states and non-uniform genetic rules, distributed randomly at the initial stage. The main genetic operator is a bit-wise partial mutation or inversion for each genetic rule. The evolutionary processes are divided into two phases, the evolving phase (A) and the test phase (B). (A) In the evolving phase, as the desirable bits are determined, an evolvable cellular classifier searches suitable genes that induce the desirable bit-patterns of the cell states. Even if the search is partially completed, all the local fitness values may be satisfied. (B) Therefore, it is required to continue to test an evolvable neural network by fixing the genes and perform parallel operations of non-uniform cellular automata. If at least one bit of the local fitness is not set during the test phase, it switches to the evolving phase in order to re-search the fittest genes. Since both processes are performed in the steady-state, the genetic changes in the transient response should be skipped by free-running for M time steps which is closely related to the size of the neighborhood. As the evolutionary processes continue, from any initial distributions of the cell states, the final cell states converge to the desired bit-patterns, and thereafter, only the genes of an evolvable cellular classifier may contribute to restoring the information.

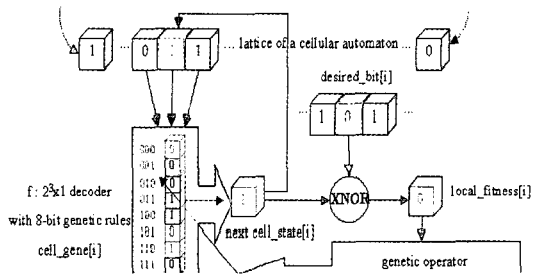


Fig. 2. Block Diagram of Evolvable Cellular Classifiers (1-D)

2.4 Programming Evolvab le Cellular Classifier

The following pseudo-coded algorithm in Table 1 shows the detailed procedure of programming an evolvable cellular classifier, which is similar to Sipper's *cellular programming*, but modified appropriately to resemble the discrete Hopfield-type memories. Because the learning algorithm is replaced with the evolutionary processes, the initial random distributions of the cell states could be classified into some desirable bit-patterns. Also, note that it is possible to switch to the evolving phase at any time during the test phase.

In the initial phase, the transient behaviors of the cell states should be preserved by fixing the genetic rules for some period of time. The propervalue of the transient period M depends on the number of neighborhood since a cell state changes by referring to those cell states in the vicinity of itself. Intuitively, we could make an

Table 1. Prgramming Evolvable Cellular Neural Network

```
Evolvable Cellular Neural Network Programming:
set all desired_bit[i];
set random_genes;
random_configuration = 0;
while not(converged),
  set random_states;
  time = 0;
  free-run CA M-time-steps with the fixed non-uniform genes;
  evaluate all local_fitness;
  if (all local_fitness = 1),  phase = test_phase;
  else phase = evolving_phase;
  end if;
  while not(all local_fitness[i] = 1),
    for all cell[i] do in parallel;
      update cell_state[i];
      if (desired_bit[i] = cell[i]),    local_fitness[i] = 1;
       else    local_fitness[i] = 0;
         if (phase = evolving_phase), mutation(gene[i]);
         end if;
      end if;
    end parallel;
    if (at_least_one local_fitness = 0), phase = evolving_phase;
    end if;
    time = time + 1;
  end while;
  random_configuration=random_configuration+1;
end while;
```

assumption that the steady-state response may be influenced in a triangular form of propagation in the nearby states. Therefore, $M > 3(2r+1)$ (1 −D case) or $M > 3\times$(the number of the neighborhood) is a good choice of preserving the transient effects at the initial stage.

2.5 Local Fitness vs. Global Fitness

Here, an interesting concept of the relationship between a local fitness and the global fitness arises, and it may establish the link from the encoded genotypes to the phenotype of feedback in the real-world environment. In our case, a simple local fitness is defined in eq.(3) which results from an error of local interaction between a cell state and the desired state, and the global fitness, in general, can be defined in eq.(4) as follows:

$$\begin{aligned} \text{global\_fitness}_i &= h(\text{local\_fitness}_i[\,\cdot\,]) \\ &= \sum_{\forall i} a_i \times \text{local\_fitness}_i[i] \end{aligned} \quad (4)$$

where h is an evaluating function, and a_i is a linear coefficient so that the global fitness may be the weighted sum of all the local fitness values.

2.6 Redundancy and Evolvable Cellular Classifiers

The local distributions of the genetic rules are redundant in a sense that the same convergent bit-patterns are likely to have many different genetic codes. The reason results from the fact that the dynamic behaviors of the same convergent bit-patterns of the cell states change with a wide variety of sets of genetic rules which eventually perform as a many-to-one mapping. Therefore, we are able to store a collection of desirable bit information under different initial conditions in the nearby Hamming distance. It means that evolvable cellular classifiers are applicable to classifiers which appropriately separate the Hamming space, in which similarcoding problems of the existing neural networks in Kosko's[11], Kang's[12] are dealt with as in associative memories. Only difference is the fact that the learning process is supervisory and evolutionary. In Table 1, the algorithm is slightly modified to include several desired bit-patterns at the same time as the evolution proceeds. That is, for each configuration, the multiple initial distributions corrupted by bit-reversal noise are applied simultaneously to the associated states with the same genetic rules. The local fitness evaluation is also changed in such a way that a local fitness value in each cell is set to 1 only if every state values of the cell at the same location for different desired values are satisfied. This is due to redundancies in producing 0's and 1's of each gene as mentioned above.

3. Main Results: Online Evolution of Evolvable Cellular Classifiers

The initial densities of the cell states are chosen so that the probability of 1's is 0.5 (p=0.5) which may vary in some applications. For example, in a pattern classification problem, the desired bits can be disturbed by random noise and these may be prepared for the initial distributions. The maximally allowed mutation probability is 100% and the mutation rate per gene is varied from 1/8 to 1/2 in both 1-D or 2-D cases. Also, the free-running period M is 15, and ai's in eq.(4) are chosen $1/L$ where L is the lattice size (L_{1D} = 64, L_{2D} = 16×16 or 32×32).

3.1 Examples of 1-D Evolvable Cellular Classifiers

Two desirable bit-patterns, 'Yin-Yang' and 'Stripes', are applied to 1-D evolvable cellular classifier. The evolving phase of the initial (0^{th}) configuration is shown in Figure 3-(a) for the 'Yin-Yang' pattern where the cell

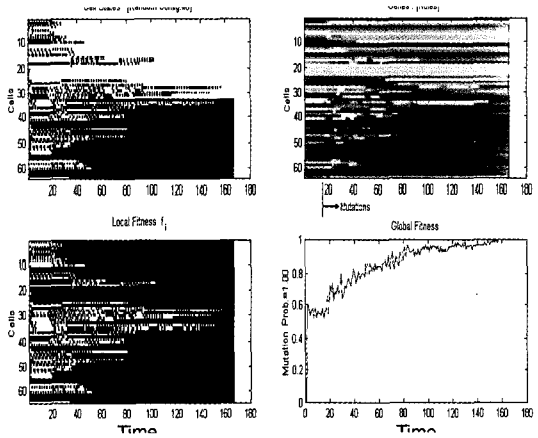


Fig. 3-(a). Evolutionary Process of the 0^{th} Configuration in 1-D(Yin-Yang)

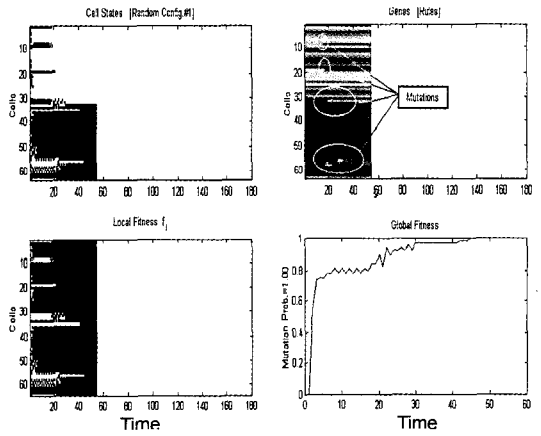


Fig. 3-(b). Evolutionary Process of the 1^{st} Configuration in 1-D (Yin-Yang)

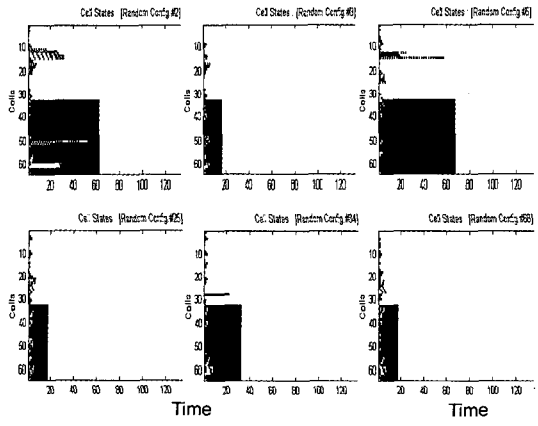


Fig. 3-(c). Evolutionary & Test Processes in 1-D Test Pattern#1 (Yin-Yang) with Random Configuration No.'s 2,3,5,25,34 and 68

states (left-top), the evolving genetic rules (right-top), the local fitness values (left-bottom), and the global fitness (right-bottom) are shown. Figure 3-(b) represents the same graphs as in Figure 3-(a), but the evolving phase of the 1^{st} configuration is shown where the frequency of mutations is reduced conspicuously. Note that the number of changes in genetic rules is decreased both in time and in size. In Figure 3-(c), it is shown that the evolving and test phases take turns in order for the genetic information to converge under the random configuration no.'s 2, 3, 5, 25, 34, & 68. Figure 4-(a) also demonstrates the evolving phase of the initial (0^{th}) configuration for the 'Stripes' pattern. Similarly, Figure 4-(b) and Figure 4-(c) show the evolving phase of the 1st configuration and the combined phases under the

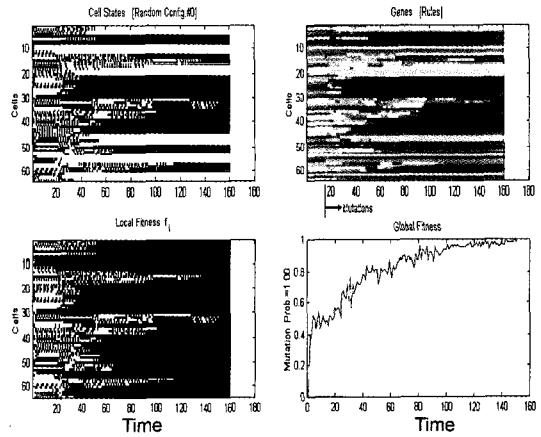


Fig. 4-(a). Evolutionary Process of the 0^{th} Configuration in 1-D (Stripes)

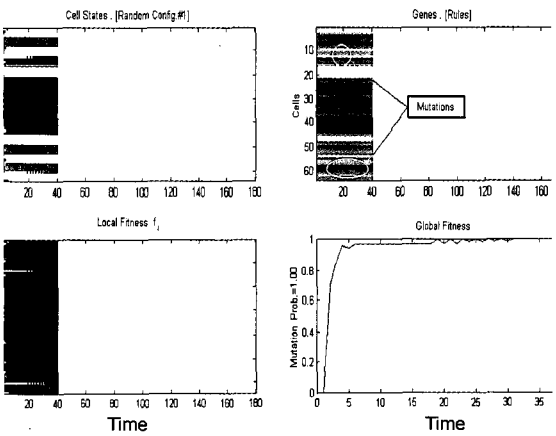


Fig. 4-(b). Evolutionary Process of the 1^{st} Configuration in 1-D (Stripes)

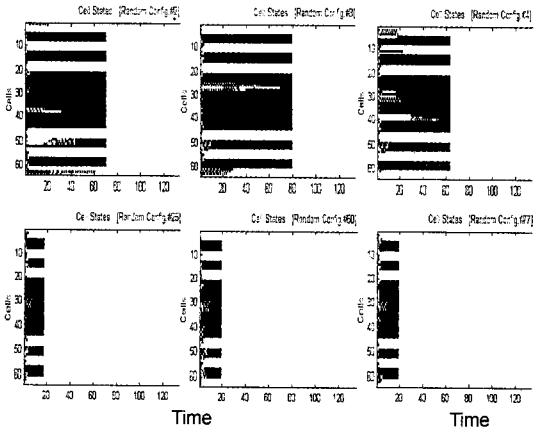


Fig. 4-(c). Evolutionary & Test Processes in 1-D Test Pattern #2(Stripes) with Random Configuration No.'s 2,3,4,25,50 and 77

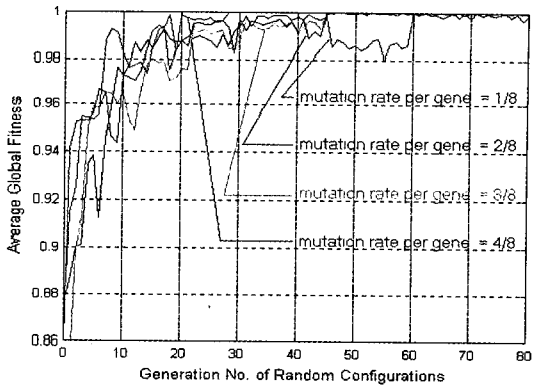
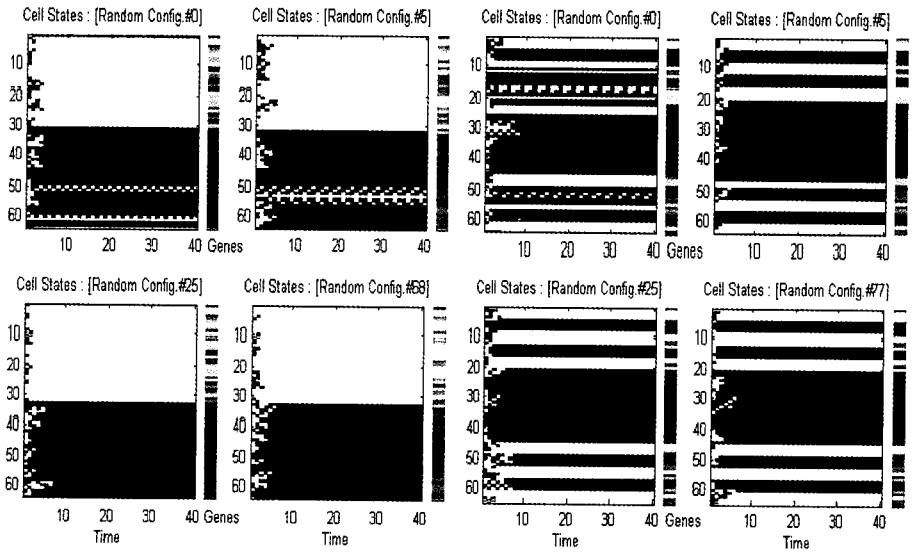


Fig. 6. Comparison of Average Global Fitnesses in Stripes

random configuration no.'s 2, 3, 5, 25, & 77, respectively. For the testing purpose only, the cell states and the associated genes are represented in Figure 5, in which the fixed genes are applied as the results of runs for each random configuration. Figure 5-(a) and Figure 5-(b) show the results of the 'Yin-Yang' test pattern and the 'Stripes' pattern, respectively. As the mutation rates per gene change, the average global fitness values of the test pattern no.2 ('Stripes') are compared in Figure 6 where the mutation rates per gene are varied 1/8, 2/8, 3/8, and 4/8. Although it is not obvious to see that a large mutation rate per gene results in better performance, the final trend reveals that the global fitness converges to 1 faster if the mutation rate per gene is larger. For a long-term evaluation of evolvable cellular classifiers, we may consider the information aspect of the entropy concept [13]. Let p be the probability of success (say, '1') in a cell (a) Config. No.'s 0,5, 25, 68(b) Config. No.'s 0,5,25,77 state and let T be the observation time, then the entropy $E(p)$ of the binomial distribution becomes

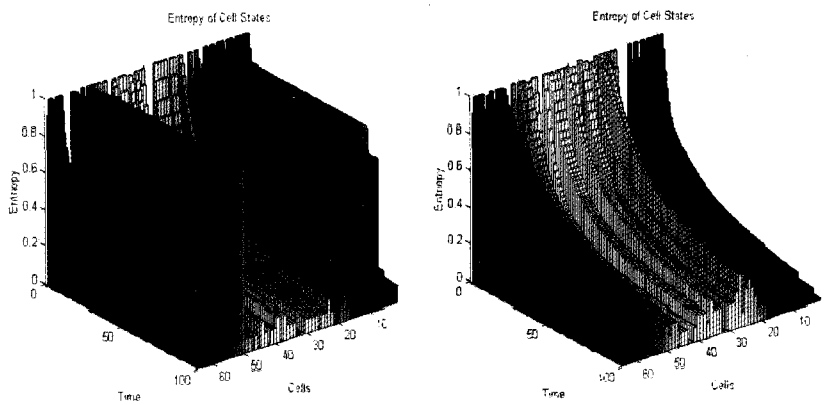
$$E[p(T)] = -p(T)\log_2 p(T) - (1-p(T))\log_2(1-p(T)) \quad (5)$$

where the probability $p(T)$ in each cell is calculated



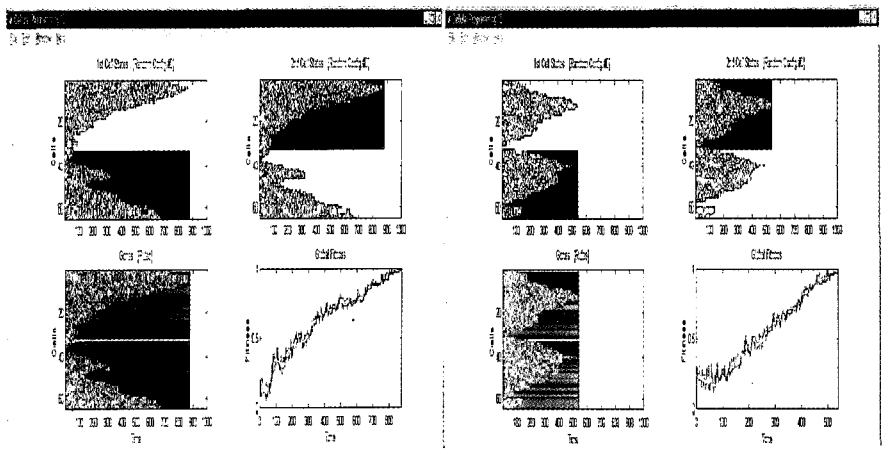
(a) Config. No.'s 0,5,25,68 (b) Config. No.'s 0,5,25,77

Fig. 5. Test Processes of Config.'s in (a) Yin-Yang (b) Stripes



(a) Cell Entropy (0^{th} Config.) (b) Cell Entropy (80^{th} Config.)

Fig. 7. Spatio-Temporal Entropy in 1-D Case



(a) Evolutionary Classification Process of the 0^{th} Configuration in 1-D Two Test Patterns (2 Yin-Yang's)

(b) Evolutionary Classification Process of the 0^{th} Configuration in 1-D Two Test Patterns (2 Yin-Yang's)

Fig. 8. 2 Cases in Evolving Phase as a Pattern Classifier

during the observation time *T*. Here, in Figure 7, the spatio-temporal entropy values are compared between two test phases in the 0^{th} and the 80^{th} configurations of a 1-Devolvable cellular classifier. The result of the 80^{th} configuration shows less entropy values, which means that the randomness in the cell structure reduces as the evolutionary phase proceeds, in accordance with our anticipation. In Figure 8, an evolvable cellular classifier plays a role as a pattern classifier to which two complementary test patterns of 'Yin-Yang' are applied where two initial conditions converge to the two desirable complementary 'Yin-Yang' patterns. It demonstrates how one set of genetic rules may produce different cell states from the various initial distributions located in the nearest Hamming distance. Here, the initial states are disturbed by 10% bit-reversal noise each. As mentioned earlier, redundancy in the cell chromosomes is the primary reason for the classification mechanism. Therefore, an evolvable cellular classifier can be used for classification problems.

3.2 Examples of 2-D Evolvable Cellular Classifiers

In 2-D evolvable cellular classifiers, a 32-bit genetic rule is included in each cell because von Neumann

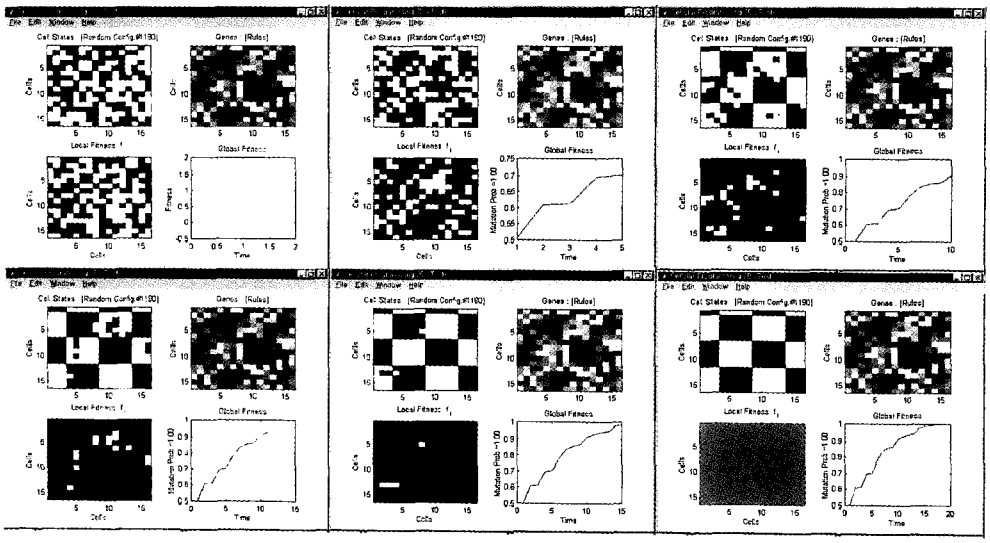
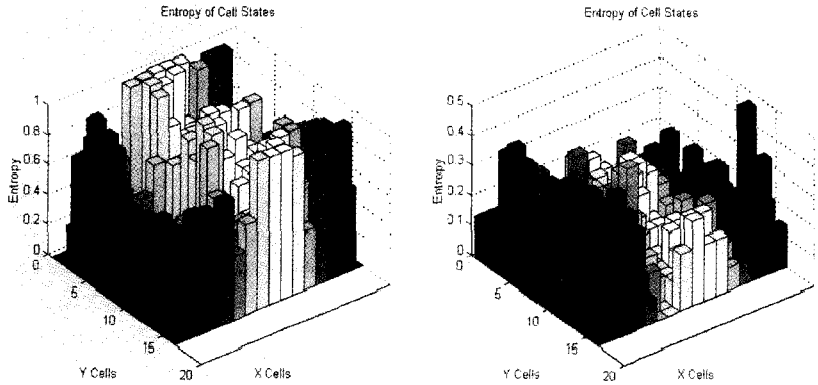


Fig. 9. Test Phase of 1190th Configuration in 2-D Evolvable Cellular Classifiers (Checker Board)



(a) Cell Entropy (0th Config.) (b) Cell Entropy (1190th Config.)

Fig. 10. Spatial Entropy of 2-D Evolvable Cellular Classifiers (at t=100, Checker Board)

neighborhood are used. Here, two test patterns, 'Checker Board' (16×16) and 'Butterfly' (32×32) are applied in the evolving and test phases. Figure 9 shows the test phases in the 1190th configuration of 'Checker Board' in which the cell states, with the fixed genes, converge to the desirable bit-pattern within 20 steps. The evolved genes show a symmetric pattern inside the grid structure. For the sake of evaluation, the entropy values at time, t = 100, are also compared between two test phases in the 0th and the 1190th Fig. 9. Test Phase of 1190th Configuration in 2-D Evolvable Cellular Classifiers (Checker Board) configurations of a 2-D evolvable cellular classifier as shown in Figure 10. As in the previous case, the spatial entropy is reduced in the later configuration due to self-organization as the evolving phases proceed. Note that as the time increases, the latter spatial entropy decreases to zero with the 1190th configuration, but the former does not. Finally, in Figure 11, the genetic rules of a complex bit-pattern 'Butterfly' are evolved and tested under 10% bit-reversal noise. Remarkably, the genetic rules in the grid structure show resemblance with the prescribed bit-patterns.

3.3 Evolvable Hardware : Implementation

We have designed and successfully tested the core

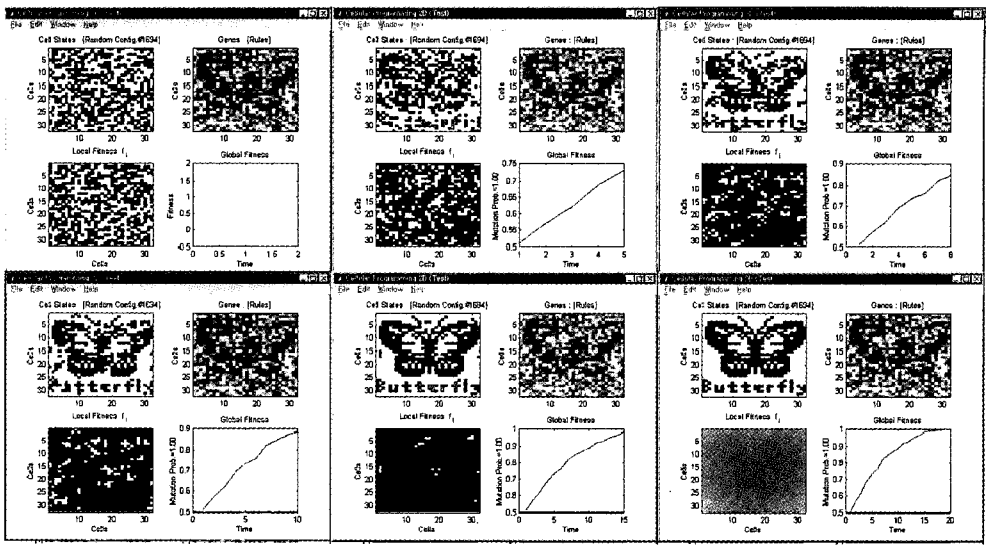


Fig. 11. Test Phase of the 289th Configuration in 2-D Evolvable Cellular Classifiers (Butterfly)

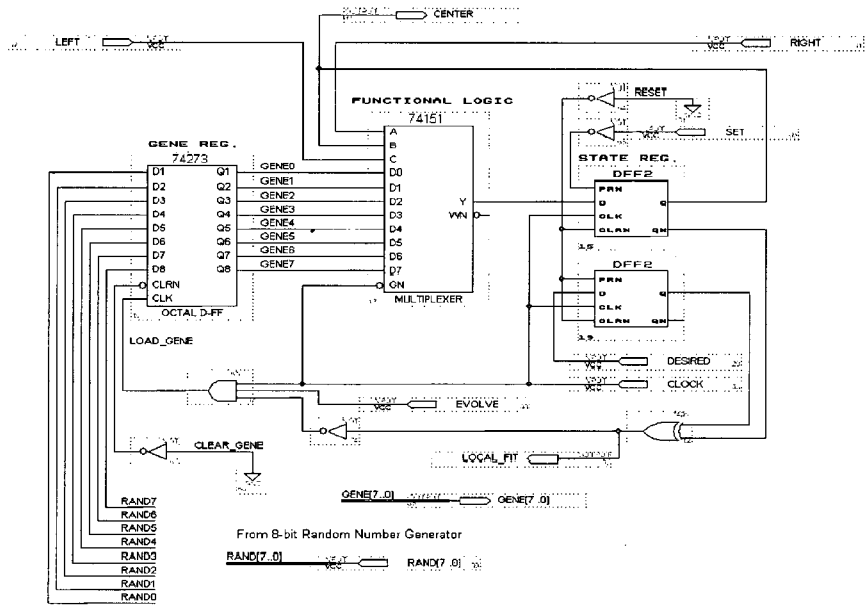


Fig. 12. Core Processing Unit of Evolvable Cellular Classifier (1-D with 3 Neighborhood)

architecture of a 1-D evolvable cellular classifier as shown in Figure 12. The genetic rules and the cell state are stored in octal D flipflops, and a D flipflop, respectively. Also, the inputs of the octal D flipflops are linked to and 8-bit random number generator. The logical function unit simply consists of an 8-to-1 multiplexer whose inputs are connected to the outputs of octal D flipflops. Moreover, the local fitness value resulted from the exclusive NOR circuit is fed to the positive-edge-triggerred enable input of the gene memory. Figure 13 represents a lattice structure consisting of 16 1-D cells with 3 neighborhood. Each block includes a random number generator with different random seeds so that it produces independent

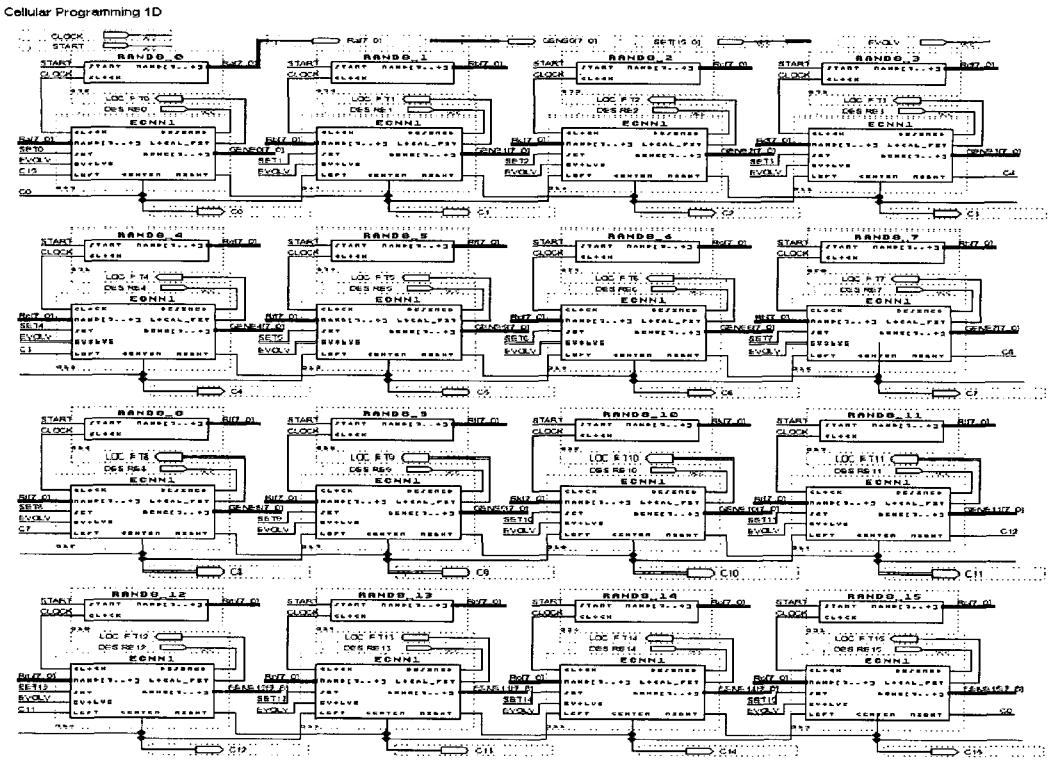


Fig. 13. An Evolvable Cellular Classifier of 16 Cells (1-D with 3 Neighborhood)

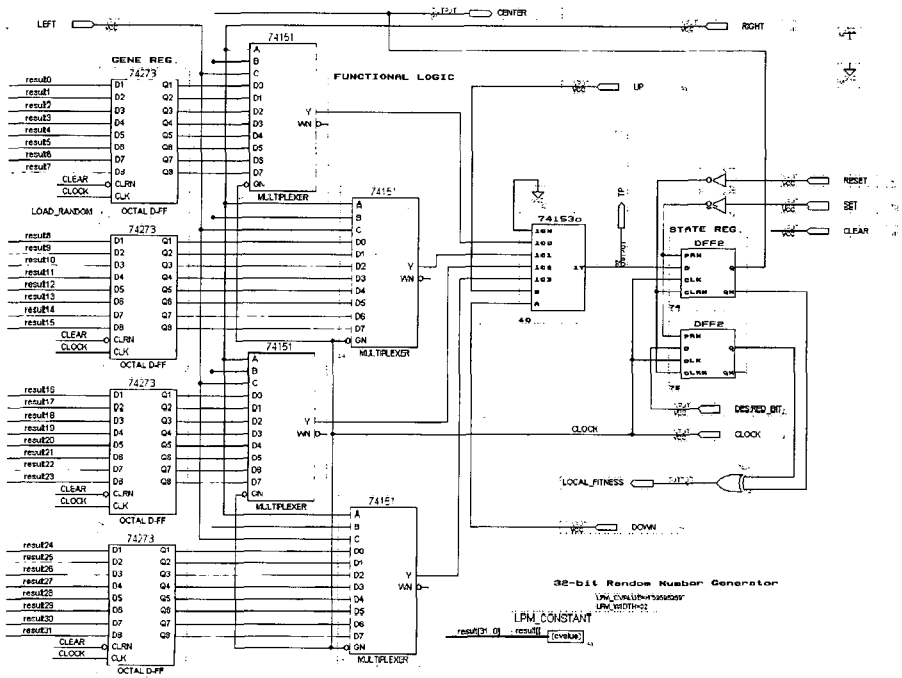


Fig. 14. Core Processing Unit of Evolvable Cellular Classifiers (2-D with 5 Neighborhood)

random integer numbers with which the mutations occur. The design of evolvable hardware is performed in a VHDL platform[14].

Also, the core processing unit of a 2-D evolvable cellular classifier is shown in Figure 14 where each cell is composed of 5 neighborhood states and a 32-bit genetic rule.

4. Conclusions and Discussion

A new paradigm of cellular classifiers, *evolvable cellular classifiers*, is proposed, and an online evolutionary algorithm is developed. Initially, the genes are evolved, and then, both evolving and test phases take turns in order to separate randomly disturbed bit-string vectors into the nearest bit-patterns. Thus, evolvable neural networks are derived in terms of non-uniform cellular automata where the nonlinear function units are replaced with the nonlinear logical and-or gates. The core processing elements refer to the neighborhood of each cell and perform self-organization of the genetic rules that eventually classify the desirable bit-patterns due to redundancies in the genetic rules. Moreover, an evolvable hardware of the proposed neuronic architecture is implemented and successfully tested. It is believed that the mechanism of evolvable cellular neuronic systems make a new promising tool for the pattern classification of bit-string information.

Acknowledgments

This paper is supported by Brain Science Research Center of Korean Ministry of Science and Technology under the grant no. 98-J04-01-01-A-07

References

[1] C. G. Langton, "Life at the edge of chaos", in *Artificial Life II*, vol.X of *SFI Studies in the Science of Complexity*, (eds. C. Taylor, J. D. Farmer, and S. Rasmussen), Massachusetts: Addison-Wesley, pp. 41-91, 1992.

[2] J. von Neumann, *Theory of Self-Reproducing Automata*, (ed. A. W. Burks), Illinois: Univ. of Illinois Press. 1966.

[3] M. Gardner, "On cellular automata, self-reproduction, the Garden of Eden and the game 'Life'", Scientific American, 1971.

[4] E. F. Codd, *Cellular Automata*, New York: Academic Press, 1968.

[5] H. deGaris, "Cam-Brain, ATR's billion neuron artificial brain project: A three year progress report", in Proceedings of IEEE 3rd Int. Conf. on Evolutionary Computation (ICEC'96), pp. 886-891, 1996.

[6] S. Wolfram, *Cellular Automata and Complexity: collected papers*, New York: Addison-Wesley, 1994.

[7] S. Levy, *Artificial Life: The Quest for a New Creation*, Random House, 1992.

[8] M. Mitchell, J. P. Crutchfield, and P. T. Hraber, "Evolving cellular automata to perform computations: Mechanisms and impediments", Physica D, Vol. 75, pp. 361-391, 1994.

[9] M. Sipper, *Evolution of Parallel Cellular Machines: The Cellular Programming Approach*, New York: Springer-Verlag., 1997.

[10] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", in Proceedings of National Academic Sciences USA, Biophysics, Vol. 79, pp. 2554-2558, 1982.

[11] B. Kosko, "Bidirectional Associative Memories", *IEEE Trans. on Systems, Man*, and Cybernetics, Vol. 18, No. 1, pp. 49-60, 1988.

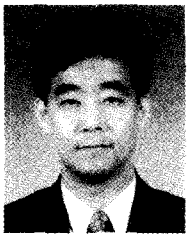
[12] H. Kang, "Multi-layer Associative Neural Networks: storage capacity vs. perfect recall", *IEEE Trans. on Neural Networks*, Vol. 5, No. 5, pp. 812-822, 1994.

[13] C. E. Shannon, "The mathematical theory of communication", The Bell System Technical Journal, Vol. 27, pp. 379-423 & pp. 623-656, 1948.

[14] Z. Navabi, *VHDL Analysis and Modeling of Digital Systems*, New York, McGraw-Hill, Inc., 1993.

주 재 호 (Jae-Ho Ju)

1997년 : 중앙대학교 제어계측공학과 공학사
1999년~현재 : 중앙대학교 제어계측학과 석사과정
주관심분야 : 셀룰라 오토마타, 신경회로망, 스테레오비젼, 퍼지 클러스터링, 퍼지 시스템 및 제어



강 훈 (Hoon Kang)

1982년 : 서울대학교 전자공학과 공학사
1984년 : 서울대학교 전자공학과 공학석사
1989년 : 미 Georgia Institute of Technology 공학박사
1989년~1990년 : 미 Georgia Institute of Technology Post Doctor
1990년~1992년 : 미 Georgia Institute of Technology 연구교수
1992년~1995년 : 중앙대학교 제어계측공학과 조교수
1995년~1999년 : 중앙대학교 전자전기공학부 부교수
2000년~현재 : 중앙대학교 전자전기공학부 교수
주관심분야 : 퍼지 시스템 및 제어, 카오스 시스템, 인공생명, 셀룰라 오토마타, 신경회로망 및 패턴인식, 진화연산, 유전 프로그래밍, 로보틱스 및 지능제어

신 윤 철 (Yoon-Cheol Shin)

1996년 : 중앙대학교 제어계측공학과 공학사
1999년~현재 : 중앙대학교 제어계측학과 석사과정
주관심분야 : 신경회로망, L-system, 퍼지 시스템 및 제어, 셀룰라 오토마타, 로보틱스 및 지능제어