The GRD Chip: Genetic Reconfiguration of DSPs for Neural Network Processing

Masahiro Murakawa, Shuji Yoshizawa, *Member*, *IEEE*, Isamu Kajitani,
Xin Yao, *Senior Member*, *IEEE*, Nobuki Kajihara, Masaya Iwata, and Tetsuya Higuchi

Abstract—This paper describes the GRD (Genetic Reconfiguration of DSPs) chip, which is evolvable hardware designed for neural network applications. The GRD chip is a building block for the configuration of a scalable neural network hardware system. Both the topology and the hidden layer node functions of a neural network mapped on the GRD chips are dynamically reconfigured using a genetic algorithm (GA). Thus, the most desirable network topology and choice of node functions (e.g., Gaussian or sigmoid function) for a given application can be determined adaptively. This approach is particularly suited to applications requiring the ability to cope with time-varying problems and real-time constraints. The GRD chip consists of a 100Mhz 32-bit RISC processor and 15 33Mhz 16-bit DSPs connected in a binary-tree network. The RISC processor is the NEC V830 which executes mainly the GA. According to chromosomes obtained by the GA, DSP functions and the interconnection among them are dynamically reconfigured. The GRD chip does not need a host machine for this reconfiguration. This is desirable for embedded systems in practical industrial applications. Simulation results on chaotic time series prediction are two orders of magnitude faster than on a Sun Ultra 2.

Index Terms—Evolvable hardware, digital signal processor, genetic algorithm, neural network, RBF network, time series prediction, nonlinear adaptive equalization

1 Introduction

Configurable hardware is an approach for realizing optimal performance by tailoring its architecture to the characteristics of a given problem. When the characteristics of a problem are known in advance and they never change in time, it is relatively easy to build configurable hardware using programmable devices like FPGAs (Field Programmable Gate Arrays) because the designer knows how the hardware should be configured.

However, for problems where designers cannot know in advance how to configure the hardware, it is required for configurable hardware to have a capability of *autonomous* and *on-line* adaptation to a given problem.

Evolvable Hardware (EHW) is a promising approach toward autonomous and on-line reconfigurable machines [1]. The basic idea of EHW is to use genetic algorithms (GAs) to find the best hardware configuration *autonomously* (i.e., without human intervention). Genetic algorithms are robust search framework where a candidate of a solution is represented as a binary bit string called a *chromosome*. In EHW, it is the hardware configuration bits that are regarded as chromosomes. Once a good chromosome is found, the EHW is reconfigured immediately according to the chromosome. When the current hardware structure of the EHW comes to be incapable of satisfying the performance goal (defined in terms of the GA's objective function), the GA is invoked to find out another hardware structure. Thus, EHW continues to configure itself autonomously in on-line fashion.

This paper describes an EHW-based chip, called a GRD (Genetic Reconfiguration of DSPs) chip. The GRD chip is designed mainly for neural network applications. It includes a 100Mhz 32-bit RISC processor and a binary-tree network of 15 DSPs. GRD chip is a building block to configure a scalable parallel processor.

In neural network applications, optimal performance for a given problem is obtained by a neural network with the most suitable topology and the most appropriate node functions. Further, to meet the time constraint imposed by real-time applications, neural network hardware systems need to be "tailored" to the size of the ideal network for the problem. In general, it is very difficult to design an optimal neural network and process it with scalable parallel hardware.

In the GRD chip, however, the GA program on the RISC processor continues to reconfigure the DSP network topology and node functions in order to adapt dynamically to the change in the characteristics of a given problem.

In addition, the GRD chip does not need the host machine control for these tasks. This is desirable for embedded systems in practical industrial applications.

- *M. Murakawa, M. Iwata, T. Higuchi are with the Computer Science Division, Electrotechnical Laboratory, 1-1-4 Umezono, Tsukuba, Ibaraki, 305-8568 Japan. E-mail: {murakawa, miwata, higuchi}@etl.go.jp.*
- *S. Yoshizawa is with the Yoshizawa Laboratory, Department of Mechano-Informatics, Faculty of Engineering, University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8656 Japan. E-mail: {murakawa, yoshi}@bios.t.u-tokyo.ac.jp.*
- *I. Kajitani is with the Hoshino Laboratory, Institute of Engineering Mechanics, University of Tsukuba, 1-1-1 Tennou-dai, Tsukuba, Ibaraki, Japan. E-mail: i_kajita@etl.go.jp.*
- *X. Yao is with the School of Computer Science, University College, University of New South Wales, Australian Defence Force Academy, Canberra, ACT, Australia 2600. E-mail: xin@cs.adfa.edu.au.*
- *N. Kajihara is with the High Performance Computing Technology Group, C&C Media Research Laboratories, NEC Corporation, 1-1, Miyazaki 4-Chome, Miyamae-ku, Kawasaki, Kanagawa 216-8555, Japan. E-mail: kajihara@ccm.cl.nec.co.jp.*

For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 109709.

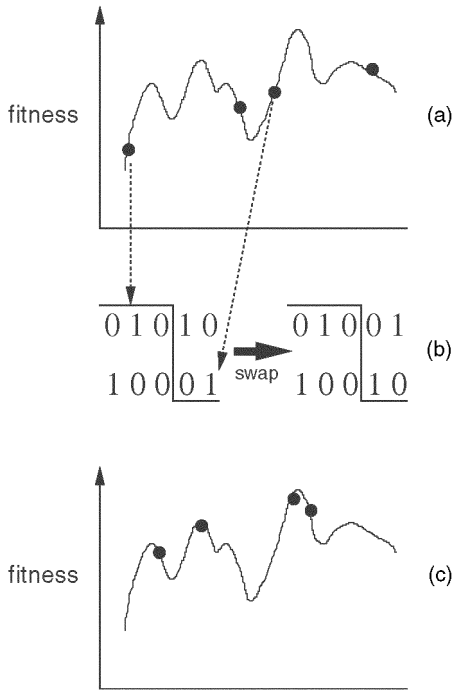

Fig. 1. Genetic algorithm. (a) Initial population. (b) Crossover. (c) Population after several generations.

In Sections 2 and 3, the idea of EHW and requirements to the adaptive neural network processing are discussed, respectively. Section 4 gives an overview how GRD chips are utilized in neural applications. Section 5 describes the GRD architecture. In Sections 6 and 7, as examples of GRD applications, chaotic time series prediction and adaptive equalization in digital mobile communications are described, respectively. Section 8 concludes this paper.

2 Evolvable Hardware

EHW is based on a genetic algorithm (GA) and software reconfigurable devices [1]. In this section, we explain the basic idea of EHW after the brief description of GA.

2.1 Genetic Algorithm

GA is a robust search algorithm which is loosely based on population genetics [2]. GA can effectively find solutions in a huge search space at a reasonable cost of computation. Before the GA search starts, candidates of solutions, represented as binary bit strings, are prepared. This is called a *population*. A candidate is called a *chromosome*. In Fig. 1a, there are four chromosomes in a population. Also, an evaluation function, called *fitness function*, needs to be defined for a problem to be solved in order to evaluate chromosomes. A chromosome with a high fitness value is likely to be a good solution of the problem.

GA search goes as follows: Two chromosomes chosen randomly from a population are mated and they go through genetic operations like the crossover to yield better chromosomes for next generations (See Fig. 1b). This is repeated until about a half of the population are replaced with new chromosomes. Because the population size is fixed, chromosomes with lower fitness values tend to be eliminated from a population. Therefore, after several generations of GA search as in Fig. 1c, relatively high fitness chromosomes remain in a population and some of them are chosen as solutions of the problem (Compare with Fig. 1a).

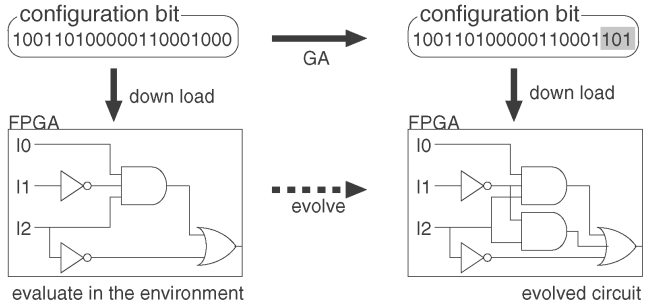

Fig. 2. Evolvable hardware at gate-level.

2.2 Basic Idea of Evolvable Hardware

The basic idea of EHW is to regard the configuration bits of a software reconfigurable device as the chromosome of GA (Fig. 2). The search space of configuration bits is very huge, but GA is very effective without a priori knowledge about the search space.

As a fitness function, we choose the performance of the hardware circuit. For example, in data compression with EHW, we use predictive function implemented by hardware [3]. As a fitness function, we chose the data compression rate. A circuit of predictive function with a higher data compression rate is likely to remain in a population. When the good chromesome is obtained, it is immediately downloaded into the reconfigurable device.

In EHW, it is not required to specify the detailed hardware design. Instead, we define a fitness function. A fitness function is the *instinct* to evolve the hardware circuit. If a fitness value of a current hardware circuit is degraded due to partial malfunction or some changes in the environment, then the GA process of EHW is invoked and the search for a better hardware is initiated. Therefore, EHW continues to reconfigure itself in order to get a better performance.

The chromosome of EHW specifies two things. One is function type of the evolution unit. In Fig. 2, the evolution units correspond to gates like AND-gate and OR-gate. The other is the interconnection among evolution units. EHW can be classified into two classes according to the grain size of an evolution unit; gate-level and function-level. Fig. 2 is an example of gate-level evolution. In function-level evolution, each evolution unit is higher hardware function than gate-level evolution [4]. The GRD chip belongs to the function-level evolution.

3 Requirements for Neural Network Processing

Most of the industrial neural network (NN) applications are limited to neural networks with *off-line* learning where learning phase and the execution (recognition) phase are separate. Such a neural network never changes during its

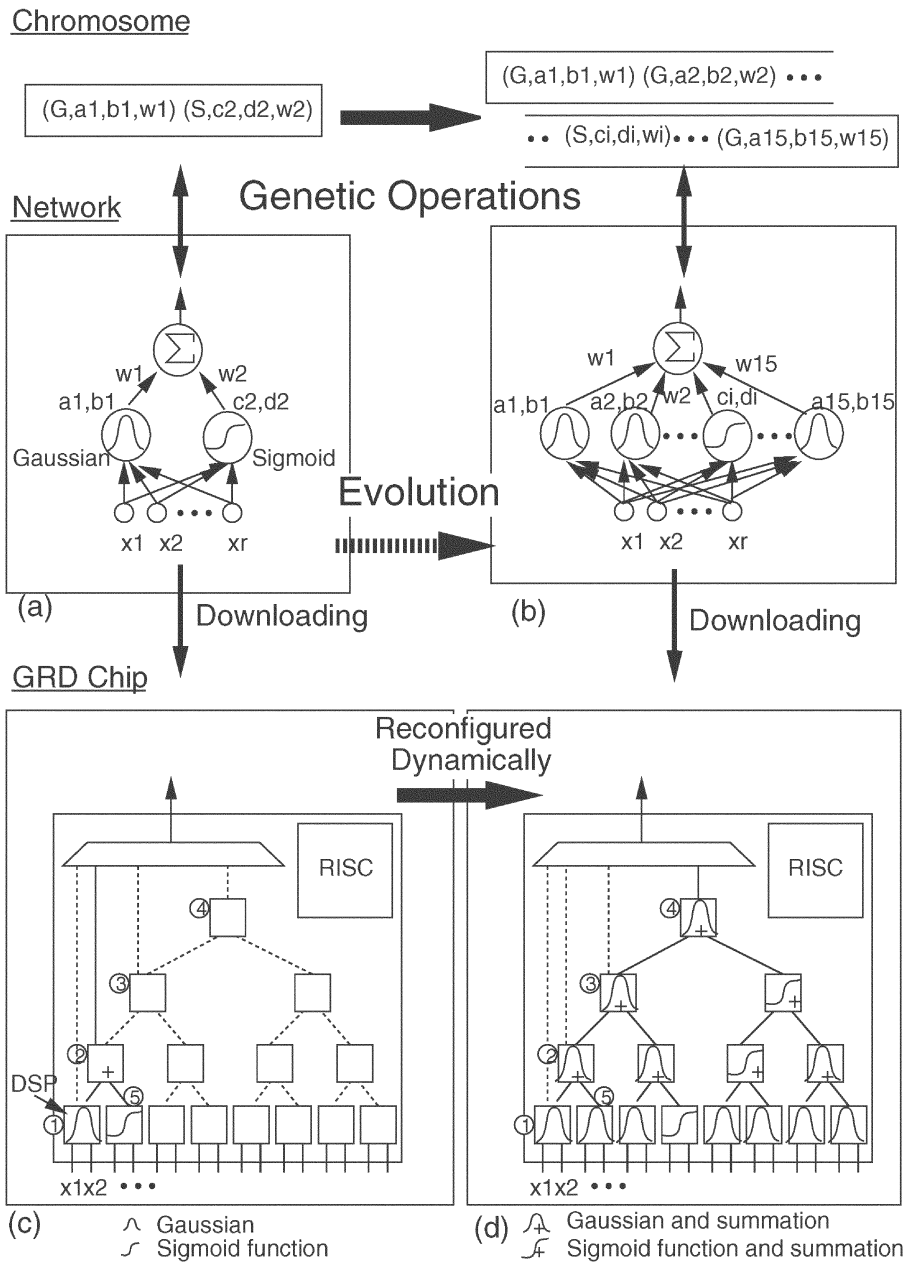

Fig. 3. Evolvable hardware for neural network learning.

execution and therefore lacks the flexibility needed. We contend that, in order to use neural networks in a broader range of practical applications, they have to be capable of *on-line* learning. On-line learning allows neural networks to adapt dynamically to changing problems. In this section, we examine this in more detail.

3.1 Ontogenic Neural Network

The advantage of a neural network is the ability to adapt to problems by changing interconnection weights on-line. However, it is very difficult to determine the topology of neural network (i.e., the number of hidden layers and units per layer) in advance of the execution. If hidden layer nodes are less than required, it is impossible for the network to learn the problem. On the other hand, if hidden layer nodes are more than required, the network overlearns the problem and, hence, has poor generalization capability. As a result, trial and error while determining the topology are inevitable to obtain optimal performance. Some neural networks, called *ontogenic neural networks*, try to solve this problem by letting the network *autonomously* determine the best topology and interconnection weights for a given problem at the execution phase [5].

Ontogenic neural networks are promising for on-line learning and are necessary for practical applications.

3.2 Lack of Dynamically Reconfigurable Neural Network Hardware

Even though an ontogenic neural network allows the dynamic adaptation to a problem, conventional neural

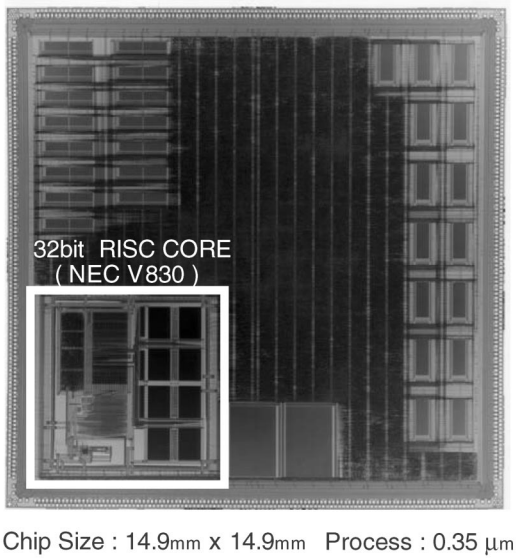

Fig. 4. GRD chip.

network hardware systems have not provided the reconfiguration capability of the network structure that is needed during execution. To our knowledge, digital neural hardware so far have been developed mainly for acceleration of neural computation [6].

Optimal performance of neural network hardware is obtained when the logical NN structure matches the physical NN hardware structure. Therefore, dynamically reconfigurable NN hardware is a key to applications of ontogenic neural networks. In addition, NN hardware systems for industrial applications should be compact for embedded systems. Stand-alone NN hardware is preferred to back-end NN hardware like SIMD neural machines.

The GRD chip is a building block for the configuration of scalable NN hardware systems. A system built with the GRD chips can dynamically reconfigure its hardware structure to be tailored to the optimal topology without a host machine.

3.3 Learning Time

Long learning time is one of the obstacles when neural networks are used to industrial applications. This is especially significant in multilayer perceptrons (MLPs) with back propagation (BP) learning. The MLP uses the sigmoid function as a node function.

To improve the learning speed, radial basis function (RBF) networks [7] may be an appealing choice. In RBF networks, the node function is a Gaussian function where the response to inputs is more localized compared with the sigmoid function. This leads to faster convergence of up to three orders of magnitude compared with MLP [8].

However, compared with MLP, RBF networks require a large number of hidden layer nodes, particularly for high-dimensional input/output spaces (the "curse of dimensionality"). This becomes an obstacle to compact implementation. Thus, a trade-off exists between learning time and size of the neural network. It was therefore our idea to mix the use of RBFs and sigmoid functions within a single architecture. Further, rather than specifying a priori how the two functions should be combined, we developed a method of using a GA to tailor the node functions in a network to a given problem adaptively. To reduce the learning time, a steepest descent method is first applied as a local learning algorithm to bring the network weights to a reasonable level. This local learning is performed in parallel by the hardware.

4 Evolvable Hardware for Neural Network Learning

As described in Section 2, we have developed 1) a learning scheme which utilizes a GA to automatically select both of the optimal network topology and the node functions, and

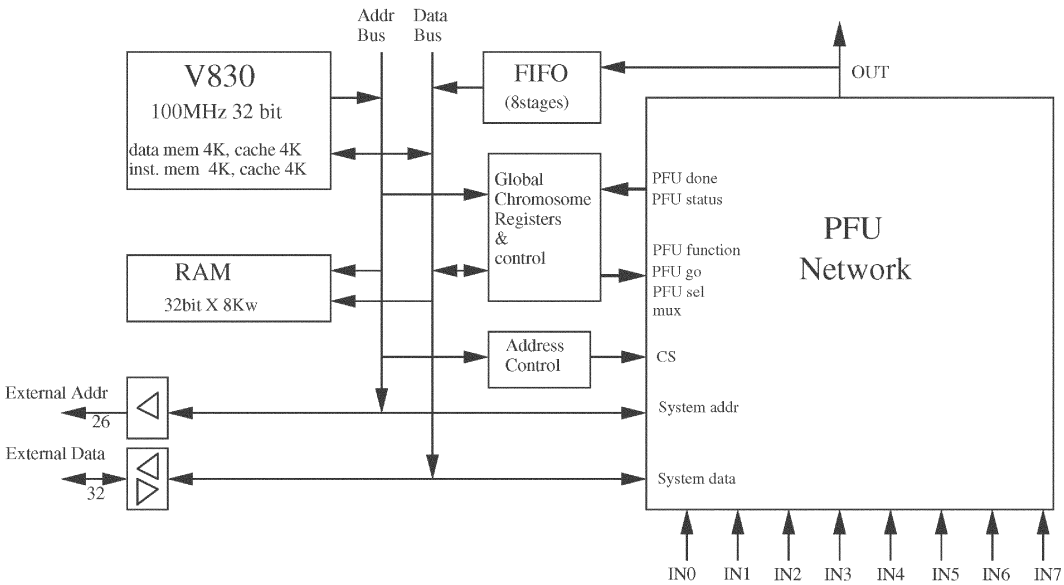

Fig. 5. Overview of the GRD chip.

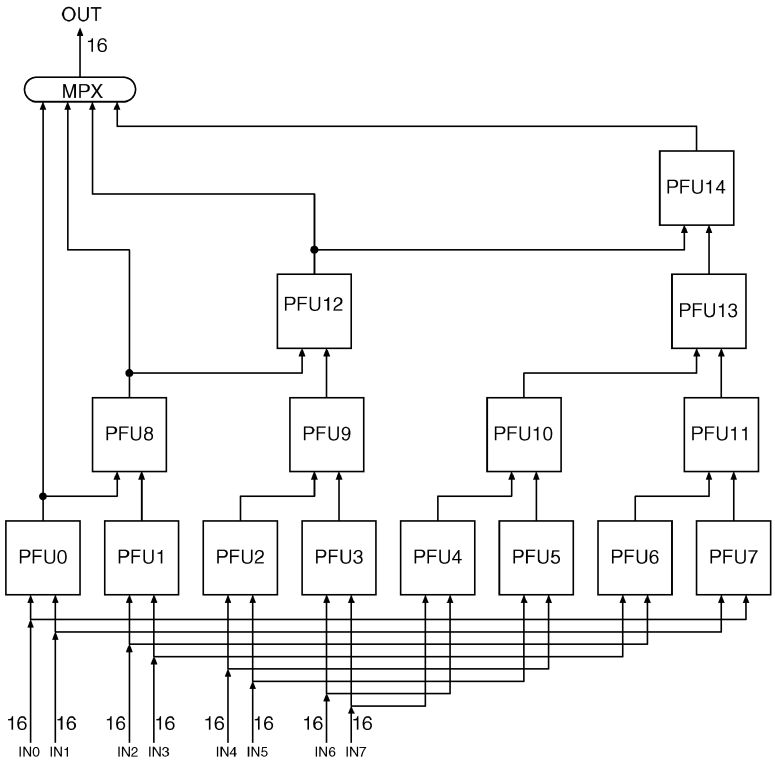

Fig. 6. PFU network.

2) the GRD chip which functions as a building block for configuring a scalable neural network hardware system.

In this section, we describe the genetic learning and then show how the network is mapped onto the GRD chips.

4.1 Genetic Learning

The neural network considered here is aimed at industrial applications which need a neural network for the approximation of nonlinear functions. The function of the network is defined as follows:

$$y = f(\boldsymbol{x}) = \sum_{k=1}^{n} w_k \mu_k(\boldsymbol{x}), \tag{1}$$

where $\boldsymbol{x} = (x_1, \cdots, x_r)$ is the input to the network, y is the output of the network and n is the number of the hidden layer node functions. For simplicity, each function $\mu_k(\boldsymbol{x})$ is either a radial basis function (RBF) or a sigmoid function:

$$\mu_k(\boldsymbol{x}) = \prod_{i=1}^{r} exp(-(x_i - a_{ik})^2 / b_{ik}) \tag{2}$$

or

TABLE 1
Comparison of Computation Time for a Node Function with Four Inputs
(9GRD Chips vs. Sun Ultra2 200MHz)

	GRD	SUN Ultra2 200MHz
execution of one RBF node	0.88 μsec	75.8 μsec
execution of one sigmoid node	0.64 μsec	74.3 μsec
execution and learning of one RBF node	3.49 μsec	688 μsec
execution and learning of one sigmoid node	2.12 μsec	661 μsec

In the GRD chip, execution of one RBF node takes $11 + 9r/2$ *cycles and learning takes* $14 + 18r$ *cycles. Execution of one sigmoid node takes* $15 + 3r/2$ *cycles and learning takes* $21 + 7r$ *cycles (*r *is the number of the inputs to the network).*

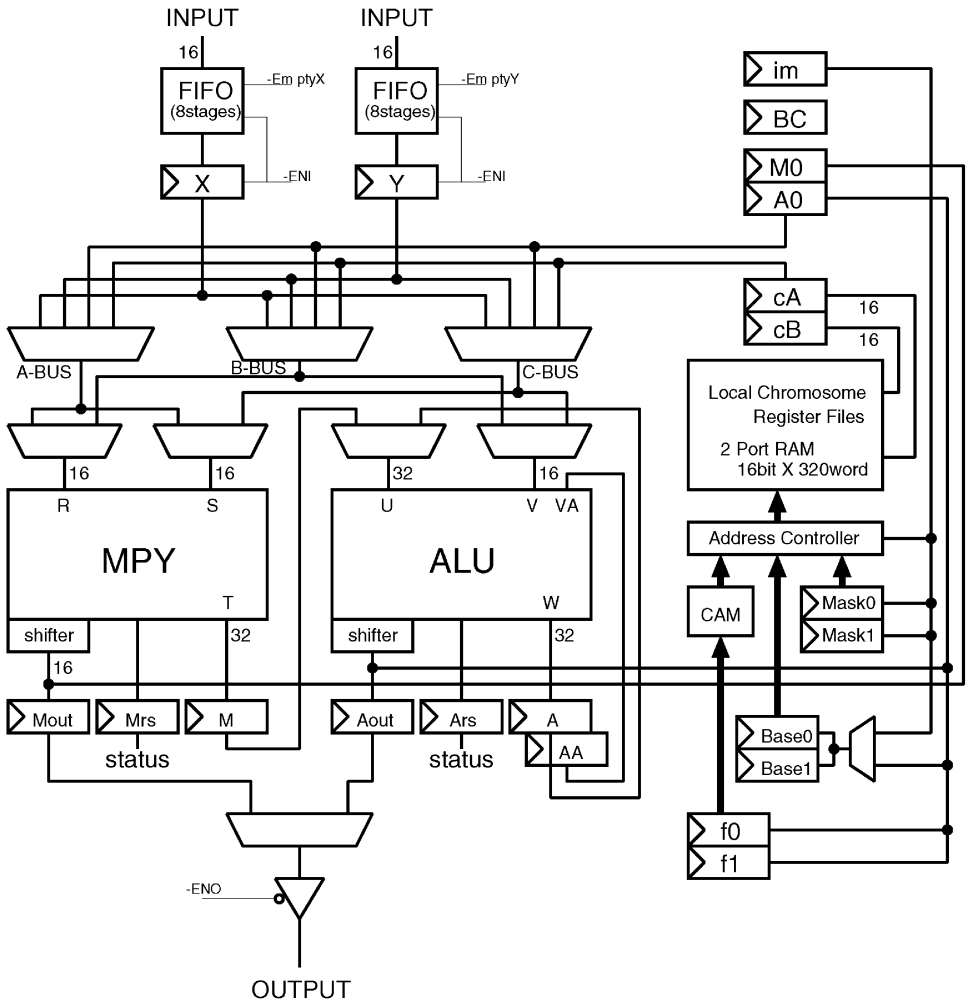

Fig. 7. PFU architecture.

$$\mu_k(\boldsymbol{x}) = \frac{1}{1 + exp(\sum_{i=1}^{r} c_{ik} x_i - d_k)}, \tag{3}$$

where r is the number of the inputs to the network. Other types of the node function can also be implemented with the GRD chip. The genetic learning determines the network topology (e.g., the number of nodes n) and the choice of node functions (e.g., Gaussian or sigmoid function) adaptively for a given application. Initial values of the weights w_k and the parameters of the node functions (e.g., $a_{ik}, b_{ik}, c_{ik}, d_k$) are also determined by the GA and then tuned by local learning with the steepest descent method (for more details of genetic operations, see [9]).

Fig. 3 illustrates this genetic learning. A chromosome of the GA represents one network. The network is evolved by applying the genetic operators to the chromosome. For example, Fig. 3 shows how a network with two hidden layer nodes (Fig. 3a) is evolved to have 15 nodes (Fig. 3b).

4.2 Mapping on the GRD Chips

Here, we show how the network obtained by the GA is mapped on the GRD chips and how they are reconfigured dynamically.

The GRD chip is a building block for the configuration of a scalable neural network hardware system. Neural network hardware of an arbitrary size can be configured with multiple GRD chips. The GRD chip has a binary tree network of 15 DSPs whose height can be reconfigured. The RISC processor in the GRD chip executes the GA. Each DSP can calculate one Gaussian or sigmoid function. Thus, one GRD chip can process 15 nodes in parallel. If more than 15

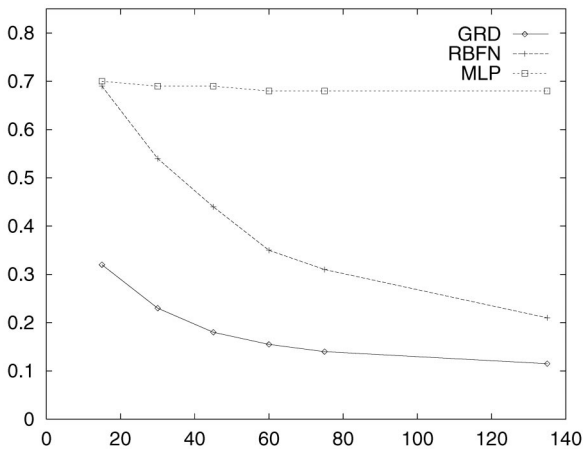

Fig. 8. Normalized prediction error versus number of hidden layer nodes.

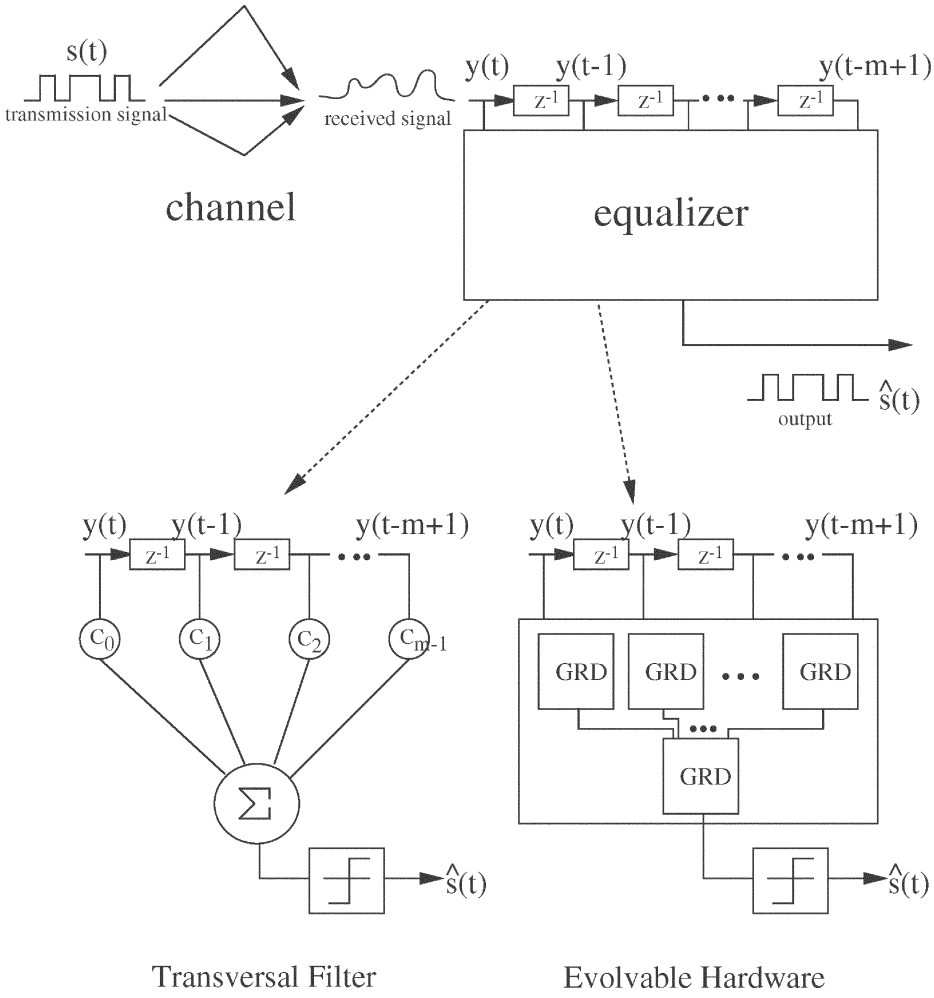

Fig. 9. Adaptive equalizers based on a linear transversal filter and based on the GRD chip.

nodes are required, GRD chips are directly connected in a tree shape. In this case, multiple chips construct a reconfigurable binary tree network of DSPs. The tree height is changed according to the chromosome of the GA.

The network structure obtained by the GA is immediately mapped on the GRD chips. For example, the network having a Gaussian function and a sigmoid function in Fig. 3a can be mapped onto the GRD chip in Fig. 3c.

The functions and tree height of the GRD chips are dynamically controlled by rewriting the chromosome on the chips. For example, in Fig. 3c, the output of the GRD chip is connected to the output of the DSP No. 2. After the evolution, in Fig. 3d, the output of the GRD chip is reconfigured to be connected to the output of the DSP No. 4. Also, in Fig. 3c, the DSP No. 5 calculates the sigmoid function. After the evolution, in Fig. 3d, this DSP is reconfigured to calculate the Gaussian.

Binary tree connections are very useful when the summation of outputs of nodes is calculated. All the DSPs in Fig. 3d are configured to conduct the summation in parallel. For example, the DSP No. 2 calculates the Gaussian first and then adds the result and the output of the DSP No. 1 and No. 5.

The above implementation is for the fastest computation. For slower applications, one GRD chip suffices for processing more than 15 nodes. A DSP in the GRD chip can process up to 84 neurons in time division multiplexing.

5 The Architecture of the GRD Chip

5.1 Overview

The GRD chip consists of a 100 Mhz 32-bit RISC processor and 15 DSPs (Fig. 4). Fig. 5 gives the overall structure of the GRD chip. The RISC processor is the NEC V830 which is designed for multimedia applications. The DSP, a 33 Mhz 16-bit fixed point processor, is called a PFU (Programmable Function Unit). 15 PFUs are connected in a reconfigurable network of a binary tree shape (See Fig. 6). The GRD chip accepts eight 16-bit inputs and generates a 16-bit output with the MIMD parallel processing of 15 PFUs. The tree shape interconnection is powerful, especially when the summation of neuron outputs is calculated.

The GRD chip includes the V830 RISC processor in order to perform genetic reconfiguration of PFUs. This means that the GRD chip can reconfigure itself. The organization of PFU is shown in Fig. 7. The content addressable memory (CAM) is employed to accelerate the computation of nonlinear functions (e.g., sigmoid function and Gaussian). Consequently the GRD chip attains 319 MCPS (Mega Connection Per Second) performance in MLP.

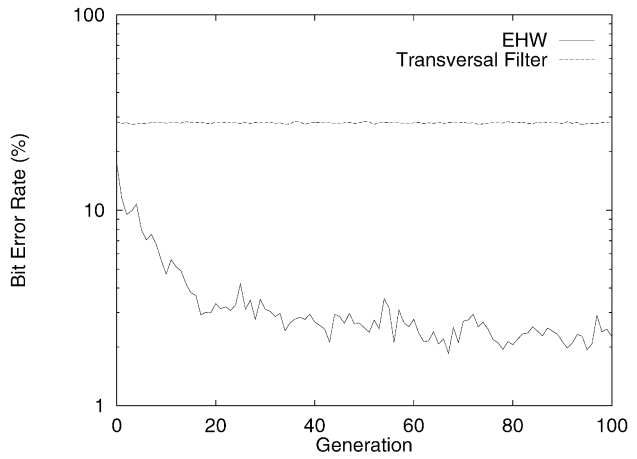

Fig. 10. Learning performance of the GRD-based equalizer (SNR: 15 dB).

The GRD chip can process up to 1,260 neurons (84 neurons per PFU). To configure a scalable hardware system easily, GRD chips can be directly connected each other via FIFO buffers inside the PFU. For example, a 19-inch rack implementation of 16 VME triple-height boards (nine GRD chips on a board) can realize the performance of 46 GCPS (Giga Connection Per Second) in MLP.

5.2 The V830 Processor

The V830 conducts the following: 1) the genetic reconfiguration of the PFU network, 2) SIMD control of the PFUs when the local learning is conducted at each PFU, and 3) master control of the other GRD chips when multiple GRD chips are used. Details of 1), 2), and 3) are described below.

5.2.1 Genetic Reconfiguration

The GA program running on a V830 determines the topology, the node functions of an optimal neural network for a given problem. The chromosomes describing the topology and the node function type of each PFU are written into *global* chromosome registers (see Fig. 5). The chromosomes for initial values of function parameters including connection weights are directly written by the V830 into *local* chromosome registers in each PFU.

The reconfiguration of the network topology corresponds to the selection of the height of the binary tree network in the GRD chip. The chromosome bits (i.e., genes) corresponding to the network topology are the selector bits of a multiplexer which actually select the tree size. At the multiplexer, one of the four PFUs (i.e., PFU No. 0, 8, 12, 14 in Fig. 6) is selected as the output of the GRD chip.

The reconfiguration time for the GRD chip is very quick. It is between 455 nsec and 461 μsec, depending on the number of neurons processed in the GRD chip. Although this is just for reference, configuration time for FPGAs is between 3 and 19 millisecond for Xilinx and Altera chips.

5.2.2 SIMD Control of 15 PFUs

After the topology and node function types are determined by the GA, weights and function parameters are tuned by the local learning. This can be done in parallel at PFUs in the GRD chip. A broadcast (BC) register and synchronization control are supplied at each PFU. With these, the V830 performs the SIMD control in local learning. Although MIMD parallel processing with PFUs assumes neural network processing, the GRD architecture can be used flexibly for other parallel processing applications (e.g., wavelet transformation).

5.2.3 Master Control of the Other GRD Chips

V830 at the top of the GRD tree is used for master control of other GRD chips in a multichip implementation. The V830 controls the other chips using nonmaskable interrupts. The chromosomes obtained by the GA program are passed to other GRD chips via external two-port FIFO buffers.

5.3 Programmable Function Unit

The PFU accepts two 16-bit inputs from other PFUs via eight stage FIFOs and generates one 16-bit output. ALU (Arithmetic Logic Unit) and Multiplier are pipelined and operate in parallel. There are four types of 32-bit instructions.

The PFU executes one function according to the chromosome in the global chromosome register. Two hundred fifty-two local chromosome registers in a PFU are used to calculate weights and function parameters. $2 \times r + 1$ local chromosome registers are needed for an RBF node (r is the number of the inputs). Therefore, up to 84 RBF nodes can be handled by a PFU. To access these registers effectively, a relative addressing mode is available.

The PFU of the GRD chip includes a CAM to speed up the calculation of nonlinear functions such as the Gaussian and sigmoid function. The PFU takes only three cycles for a Gaussian calculation.

The execution times in basic neural computation are shown in Table 1, including comparisons with SUN Ultra2 200Mhz. GRD is two orders of magnitude faster than the SUN.

6 Chaotic Time Series Prediction

The first problem to which we apply our learning method is learning the chaotic time series generated by the Mackey-Glass [10] differential equation:

$$\frac{dx(t)}{dt} = -bx(t) + a\frac{x(t-T)}{1+x(t-T)^{10}}. \quad (2)$$

This problem is recognized as a benchmark for comparing the learning and generalization abilities of different neural architectures. We compared the result of our learning method with MLP and RBF network of the same size.

6.1 Learning Methods

The learning task is to predict a value at point $x(t+I)$ from the four data in the past,

$$\{x(t), x(t-D), x(t-2D), x(t-3D)\},$$

where $D = 6$ and $I = 85$. Following previous studies [10], we generated the time series with the parameters $a = 0.2$, $b = 0.1$, and $T = 17$. Training data points were randomly selected from points t = 500-4,000 of the time series.

Each learning experiment was carried out as follows:

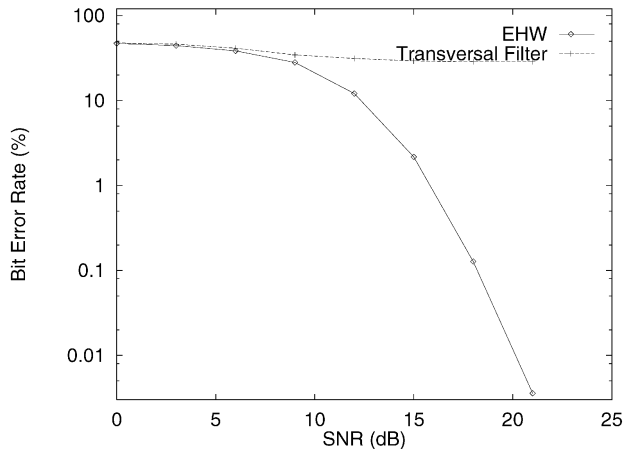

Fig. 11. Bit error rate of the GRD-based equalizer versus SNR.

6.1.1 Genetic Learning

A network to approximate the training set was evolved by genetic learning. The fitness of each network was based on the sum of the squared error over the training set. Experiments were carried out which restricted the maximum number of the hidden layer nodes to 15, 30, 45, 60, 75, and 135 nodes. The population size was 80 and each run was terminated at generation 150.

6.1.2 RBF Network

The RBF networks were produced by the k-means clustering algorithm [8]. This is the most commonly used technique for determining the structure of an RBF network.

6.1.3 Multilayer Perceptron

The three layer perceptrons were trained with the back-propagation rule. The learning rate and number of iterations were the same as that of the steepest descent method in the local learning of the GA.

6.2 Evaluation of the Network

After each network's training was complete, it was tested on a data set consisting of the next 500 time series data points following the end of the training data. The quality of the network was measured by the normalized error on this test data (where normalized error is the root-mean-squared error divided by the standard deviation of the correct prediction).

6.3 Simulation Results

The normalized prediction errors with these three learning methods are shown in Fig. 8. From Fig. 8 we can see that genetic learning can evolve a network which is superior to the MLP and the RBF network produced by the k-means clustering algorithm. For the same network size, the prediction error is roughly $1/2$.

For the learning time, the execution with nine GRD chips is almost 160 times faster than on a Sun Ultra2 200MHz. We have a simulation result that the execution with nine GRD chips takes 262 seconds, while the execution on the Sun takes 41,797 seconds.

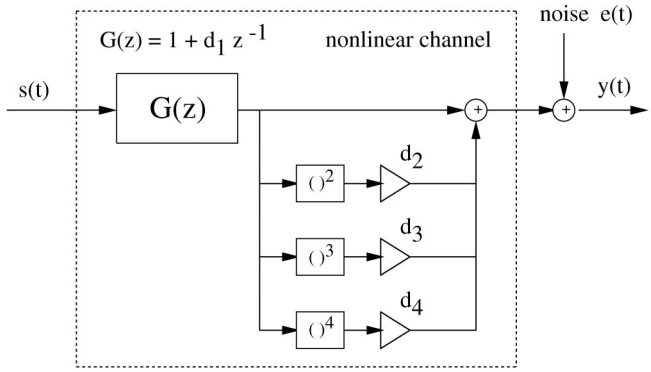

Fig. 12. Nonlinear transmission channel used in the simulations.

7 An Adaptive Equalizer for Digital Mobile Communication

To examine the performance of our system, we conducted a simulation of adaptive equalization in digital mobile communication.

High-speed communications channels are often impaired by linear and nonlinear channel distortion and additive noise. To obtain reliable data transmission in such communications systems, adaptive equalizers are required [11]. In digital mobile communications, the channel can be influenced by environmental conditions such as landscape and the presence of buildings. The task of the equalizer is to recover the transmitted symbols s(t) based on the channel observation y(t) (Fig. 9).

Existing adaptive equalization techniques for time-varying channels employ a linear transversal filter [12], [13]. However, if the nonlinear channel distortion is too severe, adaptive equalizers based on such linear transversal filters suffer from severe performance degradation. For such channels, nonlinear adaptive equalizers based on neural networks were proposed [14], [15]. But, the algorithms are very complicated for hardware implementation.

To overcome these difficulties, we apply our system to the adaptive equalizer. A communications system that employs an adaptive equalizer based on GRD chips (we call this GRD-based equalizer) is shown in Fig. 9. The transmitter sends a *known training sequence* to the receiver, and the receiver adjusts the GRD-based equalizer so that it reproduces the correct transmitted symbols.

The GRD-based equalizer has three advantages. First, the execution speed of the equalizer is extremely fast because the result of adaptation to the environment is the hardware structure itself. Second, the GRD-based equalizer can accomplish on-line adaptation. Third, the nonlinear functions of the hidden nodes make possible nonlinear equalization.

7.1 Learning Performance of the GRD-Based Equalizer

We simulated the learning performance of the proposed GRD-based equalizer. The transfer function of the channel was given by $G(z) = 1.0 + 1.5z^{-1}$ and zero-mean white Gaussian noise was added to the output of the channel. The order of the equalizer was 2 (m = 2).

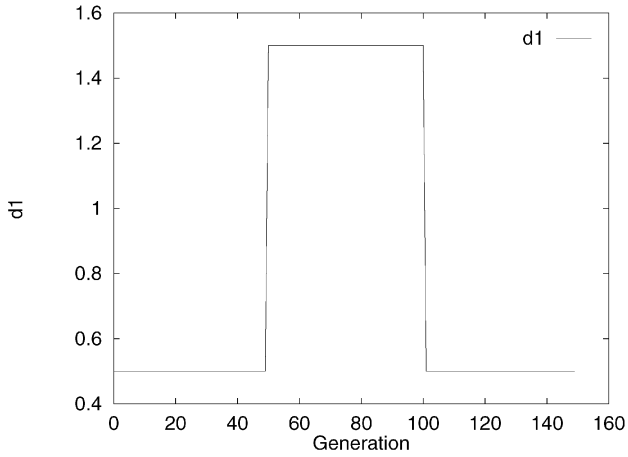

Fig. 13. Time-varying channel (drastic change in d_1).

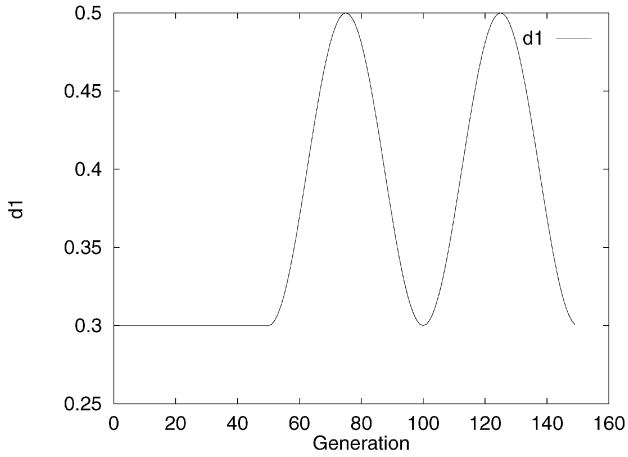

Fig. 14. Time-varying channel (gradual change in d_1).

A training set of eight data points was generated at every generation. Using a population size of 80, the fitness of each individual was determined by $n/8$, where n was the number of correct classifications by the GRD chip. The bit-error-rate (BER) was defined as the ratio of misclassified to correct symbols in the output of the best-of-generation individual. The BER was evaluated at every generation based on 10^5 random input symbols. Simulations were carried out which restricted the maximum number of the hidden layer nodes to 15.

Fig. 10 shows the learning performance of this simulation for a signal-to-noise ratio (SNR) of 15 dB. The solid curve was obtained by averaging the results of 100 independent runs. The broken line shows the learning curve of a transversal-filter-based equalizer, whose total number of training sequences was the same as that of the GRD-based equalizer. As can been seen, the BER of the GRD-based equalizer is far lower. This is due to the ability of the evolved network to synthesize nonlinear functions.

Fig. 11 shows the BER versus SNR achieved by the GRD-based equalizer at generation 100. We found that a significant improvement in the BER could be achieved by the GRD-based equalizer, especially at a high SNR.

7.2 Adaptive Equalization of Time-Varying Channels

In real communications systems, the characteristics of the channel are usually time-varying. Hence, adaptive equalizers are required to follow such changes and compensate for the channel distortion.

We therefore simulated the performance of the GRD-based adaptive equalizer for time-varying channels, using the nonlinear channel shown in Fig. 12. The transmitted sequence is passed through a linear channel whose transfer function is $G(z) = 1 + d_1 z^{-1}$ and the output of the channel is added to the nonlinear harmonics. The value of the gain coefficients d_2, d_3, and d_4 determines how severe the nonlinear distortion will be. Such nonlinear channel models are frequently encountered in data transmission over digital satellite links. The linear transversal-filter-based adaptive equalizer cannot compensate for such nonlinear channel distortion.

We simulated the bit-error-rate (BER) achieved by the GRD-based adaptive equalizer whose order m was 2. Simulations were performed for the case in which d_1 changed drastically during evaluation (Fig. 13) and for the case in which d_1 changed gradually (Fig. 14). In the

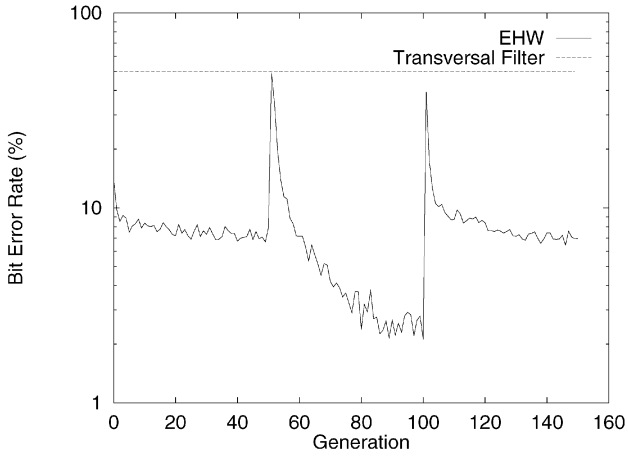

Fig. 15. Adaptive equalization of the GRD-based equalizer (SNR: 15 dB).

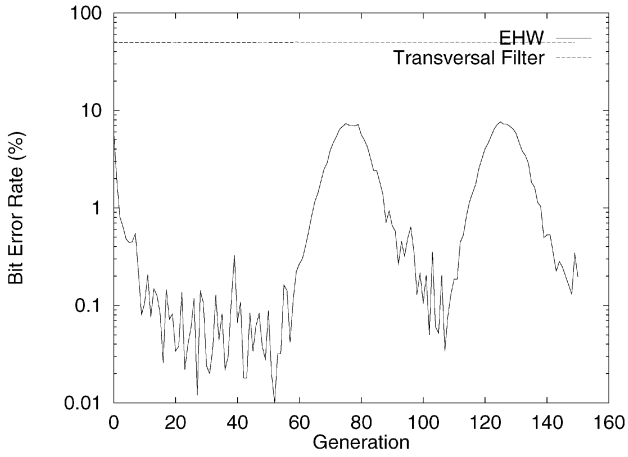

Fig. 16. Adaptive equalization of the GRD-based equalizer (SNB: 15 dB).

simulations, the coefficients were set to $d_2 = 0.6$, $d_3 = 0.5$, and $d_4 = 0.4$. The length of the training sequence was 8. For the genetic learning, the maximum number of the hidden layer nodes was restricted to 15 and the population size was 80. The results were averaged over 100 independent runs.

The BER achieved by the GRD-based equalizer for the channels of Fig. 13 and Fig. 14 are shown in the graphs of Fig. 15 and Fig. 16, respectively. These graphs demonstrate that GRD-based equalizers have the ability to follow both drastic and gradual environmental change.

For the learning of adaptive equalizer, we have a simulation result that the execution of 150 generations with one GRD chip takes 2.51 seconds while the execution on Sun Ultra2 200MHz takes 36.87 seconds.

8 Conclusion

We have described the GRD chip which is an evolvable hardware chip for neural network processing. The GRD chip realizes 1) autonomous configuration of hardware structure by genetic algorithm and 2) on-line hardware configuration. With these features, the GRD chip can deal with practical industrial applications, especially those which require the ability to cope with time-varying problems and real-time constraints. In addition, GRD is suitable for embedded use and also for multichip configuration.

The GRD chip was just manufactured in April 1998. It is planned to use it for two applications, CATV modem and prosthetic EMG(Electro Myo Graph)-controlled hand. Those are challenging applications with time-varying nature and real-time constraint to demonstrate the GRD chip.

Acknowledgments

This work is supported by MITI Real World Computing Project (RWCP). We thank Dr. Otsu and Dr. Ohmaki in Electrotechnical Laboratory and Dr. Shimada in RWCP for their support.

References

[1] T. Higuchi, T. Niwa, T. Tanaka, H. Iba, H. Garis, and T. Furuya, "Evolvable Hardware with Genetic Learning," *Proc. Simulation of Adaptive Behavior,* 1992.
[2] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison Wesley, 1989.
[3] H. Sakanashi, M. Salami, M. Iwata, S. Nakaya, T. Yamauchi, T. Inuo, N. Kajihara, and T. Higuchi, "Evolvable Hardware Chip for High Precision Printer Image Compression," *Proc. 15th Nat'l Conf. Artificial Intelligence (AAAI98),* 1998.
[4] M. Murakawa, S. Yoshizawa, I. Kajitani, T. Furuya, M. Iwata, and T. Higuchi, "Hardware Evolution at Function Level," *Proc. Parallel Problem Solving from Nature IV,* pp. 62-71, 1996.
[5] E. Fiesler, "Comparative Bibliography of Ontogenic Neural Networks," *Proc. Int'l Conf. Artificial Neural Networks,* pp. 793-796, 1994.
[6] *CNAPS Server, Preliminary Data Sheet.* Adaptive Solutions, Inc., 1992.
[7] M.J.D. Powell, "Radial Basis Functions for Multivariable Interpolation: A Review," *Algorithms for Approximation,* M.G. Cox, ed., pp. 143-167, 1987.
[8] J. Moody and C.J. Darken, "Fast Learning in Networks of Locally-Tuned Processing Units," *Neural Computation,* no. 1, pp. 281-294, 1989.
[9] M. Murakawa, S. Yoshizawa, I. Kajitani, and T. Higuchi, "On-Line Adaptation of Neural Networks with Evolvable Hardware," *Proc. Seventh Int'l Conf. Genetic Algorithms,* pp. 792-799, 1997.
[10] M.C. Mackey and L. Glass, "Oscillation and Chaos in Physiological Control Systems," *Science,* vol. 197, pp. 287-289, 1977.
[11] J. Proakis, *Digital Communications.* Prentice Hall, 1988.
[12] B. Widrow, *Adaptive Signal Processing.* Prentice Hall, 1985.
[13] S.U.H. Qureshi, "Adaptive Equalization," *Proc. IEEE,* vol. 73, pp. 1,349-1,387, 1985.
[14] S. Chen, G.J. Gibson, F.N. Cowan, and P.M. Grant, "Adaptive Equalization of Finite Non-Linear Channels Using Multilayer Perceptrons," *Signal Processing,* vol. 20, no. 2, pp. 107-119, 1990.
[15] S. Chen, G.J. Gibson, F.N. Cowan, and P.M. Grant, "Reconstruction of Binary Signals Using an Adaptive Radial Basis Function Equalizer," *Signal Processing,* vol. 22, no. 1, pp. 77-93, 1991.

Masahiro Murakawa received his BE, ME, and PhD degrees in mechano-informatics engineering from the University of Tokyo in 1994, 1996, and 1999, respectively. He is currently a researcher at the Electrotechnical Laboratory, Tsukuba, Japan. His research interests include evolutionary algorithms, reconfigurable computing, neural networks, and reinforcement learning. He received the best paper award at the second international conference on evolvable systems. He is a member of the Information Processing Society of Japan (IPSJ) and Japanese Neural Network Society (JNNS).

Shuji Yoshizawa received BS, MS, and PhD degrees in engineering in 1962, 1964, and 1971 from the University of Tokyo. From 1974 to 1992, he was an associate professor in the Department of Mathematical Engineering, University of Tokyo. He is currently a professor in the Department of Mechano-Informatics, University of Tokyo. His research interests are mainly in nonlinear dynamical systems, neural networks, and MEG measuring and modeling of brain functions. He is the president of Japan Neural Network Society, a member of the IEEE, INNS, ENNS, SICE, IEICE, JMES, and JSME.

Isamu Kajitani received his BE and ME degrees from the College of Engineering Sciences, University of Tsukuba in 1994 and 1996, respectively. Since 1996, he has been working toward the PhD degree in evolvable hardware at the same university. His research interests include evolutionary algorithms, reconfigurable computing, neural networks, and reinforcement learning. He received the Best Student Paper award at the Second International Conference on Evolvable Systems.

Xin Yao received his BSc from the University of Science and Technology of China (USTC) in 1982, his MSc from the North China Institute of Computing Technologies (NCI) in 1985, and his PhD from USTC in 1990. He is an associate professor in the School of Computer Science, University College, the University of New South Wales (UNSW), Australian Defence Force Academy (ADFA). He held post-doctoral fellowships at the Australian National University (ANU) and the Commonwealth Scientific and Industrial Research Organisation (CSIRO) before joining ADFA in 1992. He has published a number of papers in the fields of evolutionary computation and neural networks. He was the program committee co-chair of IEEE ICEC '97 in Indianapolis and CEC '99 in Washington, D.C. He was also a program committee chair/co-chair of the Third International Conference on Computational Intelligence and Multimedia Applications (ICCIMA '99), the Second Asia-Pacific Conference on Simulated Evolution And Learning (SEAL '98), ICCIMA '97, SEAL '96, and the Eighth Australian Joint Conference on AI (AI '95). He is a senior member of the IEEE, an associate editor of *IEEE Transactions on Evolutionary Computation* and *Knowledge and Information Systems: An International Journal*, and an editorial board member of *Journal of Cognitive Systems Research*. He is the second vice-president of the Evolutionary Programming Society.

Nobuki Kajihara received his BE degree in electronics engineering from Yamaguchi University in 1981 and ME degree in control engineering from Osaka University in 1983. He is a principal researcher at NEC C&C Media Research Laboratories, which he joined in 1986 and where he has been engaged in the research and development of parallel circuit simulation machine, parallel neural network simulation machine, and recofigurable architecture. His research interests include reconfigurable computing, parallel architectures, and neural networks. He is a member of the Information Processing Society of Japan, the Institute of Electronics Information and Communication Engineers, Japan Society for Artificial Intelligence, and Japan Neural Network Society.

Masaya Iwata received his BE, ME, and PhD degrees in applied physics from the Osaka University in 1988, 1990, and 1993, respectively. He is a senior researcher in the Computer Science Division of Electrotechnical Laboratory, AIST, MITI, Japan. His research interests are in developing adaptive hardware devices using genetic algorithms and their applications to pattern recognition, etc. He was a postdoctoral fellow in optical computing at ONERA-CERT, Toulouse, France, in 1993.

Tetsuya Higuchi received BE, ME, and PhD degrees, all in electrical engineering, from Keio University in 1978, 1980, and 1984, respectively. He heads the Evolvable Systems Laboratory in Electrotechnical Laboratory, AIST, MITI, Japan. He is also in charge of the adaptive devices group in the MITI national project, Real World Computing Project. His current interests include evolvable hardware systems, parallel processing architecture in artificial intelligence, and adaptive systems.