Real-World Applications of Analog and Digital Evolvable Hardware

Tetsuya Higuchi, Masaya Iwata, Didier Keymeulen, Hidenori Sakanashi, Masahiro Murakawa, Isamu Kajitani, Eiichi Takahashi, *Member, IEEE,* Kenji Toda, Mehrad Salami, Nobuki Kajihara, and Nobuyuki Otsu

***Abstract*— In contrast to conventional hardware 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. This capacity for adaptation, achieved by employing efficient search algorithms based on the metaphor of evolution, has great potential for the development of innovative industrial applications. This paper introduces EHW chips and six applications currently being developed as part of MITI's Real-World Computing Project; an analog EHW chip for cellular phones, a clock-timing architecture for Giga hertz systems, a neural network EHW chip capable of autonomous reconfiguration, a data compression EHW chip for electrophotographic printers, and a gate-level EHW chip for use in prosthetic hands and robot navigation.**

***Index Terms*—Evolutionary computation, evolvable hardware, prosthetics, robot navigation.**

I. Introduction

ONE OF the most remarkable features of the human brain is its plasticity. It is the plasticity of this natural neural network that makes it capable of flexible and robust information processing. This is in sharp contrast to conventional computer hardware where all the circuits and interconnections are fixed, making adaptive change impossible. Research into the development of adaptive machines is essentially seeking to reduce this contrast by incorporating plasticity within machines. The motivation behind this approach is to develop machines that can change their own hardware structure to adapt to new environments, and thus provide better performance.

There has been significant research progress in hardware-based adaptive machines, such as perceptrons, WISARD [11], and ALN (adaptive logic network) [14]. Hardware adaptation, however, has been limited in the sense that adaptation is offline. This means that, in practice, such systems are only used after the adaptation or the learning phase is complete. Ideally, however, adaptive machines should be capable of changing the structure of their hardware while they are operating in real-time (i.e., online adaptation).

Evolvable hardware (EHW) is a new concept in the development of online adaptive machines. In contrast to conventional hardware where the structure is irreversibly fixed in the design process, EHW is designed to adapt to changes in task requirements or changes in the environment through its ability to reconfigure its own hardware structure online (dynamically) and autonomously. This capacity for adaptation, achieved by evolutionary algorithms (such as genetic algorithms (GA's) [2], [16]), has great potential for the development of innovative industrial applications.

Although the concept of EHW is relatively new, some EHW applications are already being evaluated for their commercial value. In this paper, we introduce five EHW chips and their practical applications. These are currently being developed as part of Ministry of International Trade and Industry's (MITI's) Real-World Computing Project: a general-purpose EHW chip (to be used in prosthetic hands and robot navigation), a data compression EHW chip for electrophotographic printers, a neural network EHW chip capable of autonomous reconfiguration, an analog EHW chip for cellular phones, and an EHW-based clock-timing architecture.

Although important classifications of evolvable hardware and bio-inspired machines have already been proposed [4], [10], in this paper a basic distinction is made between digital-hardware evolution and analog-hardware evolution. Digital hardware evolutions may be further classified into gate-level evolution and function-level evolution.

Much of the research on EHW has focused on gate-level evolution: the evolution of digital-hardware involving primitive hardware gates, such as AND-gates and OR-gates. As two examples of what is possible with gate-level digital hardware evolution, we shall introduce 1) a logic circuit that can adaptively synthesize for noise-insensitive pattern recognition, such as in a neural network and 2) a fault-tolerant control circuit capable of adapting to environmental changes.

These are demonstrated in the following applications. In a prosthetic hand control application, evolvable hardware is used to implement a pattern recognition circuit tailored to the individual user. The hand is operated using electromyograph (EMG) signals from the skin surface. The mapping of EMG patterns to desired hand actions is done with the EHW-based pattern recognition circuit, which is capable of noise-insensitive recognition. Because EMG patterns have strong individual differences and may change due to physical body

Manuscript received January 22, 1999; revised May 28, 1999. This work was supported by the MITI Real World Computing Project (RWCP).

T. Higuchi, M. Iwata, D. Keymeulen, H. Sakanashi, M. Murakawa, I. Kajitani, E. Takahashi, K. Toda, and N. Otsu are with the Electrotechnical Laboratory, Ibaraki, 305-8568 Japan.

N. Kajihara is with NEC Laboratory, Kawasaki, 216-0033 Japan.

M. Salami is with Swinburne University of Technology, Hawthorn 3122, Australia.

Publisher Item Identifier S 1089-778X(99)07198-2.

conditions, hardware designers cannot design such circuits in advance using the conventional approach to hardware design, where design specifications must be identified early in the design process. This is a typical example of where the hardware is required to be adaptive.

As a second application of gate-level evolution, mobile robot navigation is described. In the robot, the EHW works as a Boolean controller that accepts sensory inputs and generates wheel motor commands. The task is to track a moving target represented by a colored ball, while avoiding obstacles during its motion in a nondeterministic and nonstationary environment. For example, even if some of the sensors of the robot are intentionally broken, the controller on the EHW is reconfigured to continue tracking the ball with the remaining functional sensors. This demonstrates how EHW can adapt to environmental changes effectively. After discussing these two applications, we shall describe the digital EHW chip, which includes GA hardware, reconfigurable logic gates, and a CPU core.

A data compression chip for electrophotographic printers is also described as a further example of more dedicated gate-level EHW chips. The data compression algorithm used in this chip is based on predictions made for image data to be sent to the printer. The accuracy of the predictions is a major factor in determining data compression ratios, since higher ratios can be obtained when the prediction is more precise. Prediction accuracy varies according to the data content, however, and thus this needs to be tailored to the specific data to be compressed. In this application, EHW is used as adaptive prediction hardware for efficient data compression, making it possible to attain compression ratios that are double those obtainable with the existing international standards in lossless data compression. Furthermore, because decompression is also performed quickly with the EHW, it is possible to satisfy the fast-printing-speed requirements of electrophotopgraphic printers.

In function-level evolution, hardware synthesis involves higher-level hardware functions than the primitive gates (e.g., AND-gates) of gate-level evolution. With function-level evolution, it is possible 1) to synthesize more useful hardware functions and 2) to design larger hardware circuits than is possible with gate-level evolution.

An autonomously reconfigurable neural network chip is described as an example of function-level EHW. In neural network processing, the tasks of determining the topology of the network and assigning appropriate functions to the nodes heavily influence learning times and learning performance. As yet, however, there is no solid theoretical basis for determining both topology and node functions. The GA in the chip can determine both of the topology and node functions that are most suitable to a given problem.

In contrast to digital-hardware evolution, analog-hardware evolution is a relatively new concept. This new paradigm within EHW is growing rapidly and is very close to industrialization. As an example of this, an analog EHW chip for cellular phones is described. Although the chip works in offline evolution mode, this approach has two advantages: 1) high yield rates and 2) smaller circuit size, leading to lower power consumption and lower manufacturing costs.

A second example of analog hardware evolution is a clock-timing architecture for very high-speed digital systems, which are close to gigahertz speeds. In such high-speed large scale integrations (LSI's), precision in clock distribution is extremely difficult to maintain, leading to poor yield rates. The EHW-based clock architecture, which has been implemented in a test chip, enables genetic adjustments in timing to be made throughout the chip.

This paper is organized as follows. Section II describes the basic concepts of EHW. Section III details two applications of gate-level evolution and a general-purpose digital EHW chip. In Section IV, a special-purpose EHW chip for data compression is discussed as an example of the industrial use of EHW. As an example of function-level evolution, Section V describes an EHW-based neural network chip. Section VI provides details of the analog EHW chip for cellular phones. Section VII describes the EHW-based clock timing architecture, followed by the conclusion.

II. Evolvable Hardware: Basic Concepts

Evolvable hardware is based on the idea of combining reconfigurable hardware devices with evolutionary algorithms, such as GA's [4], [6]. There are a variety of ways in which reconfigurable hardware and evolutionary algorithms can be combined, depending on the purposes of the evolved hardware.

When innovative solutions to difficult hardware design problems are required [26], genetic programming may be used to create hardware design in terms of hardware description languages (HDL's) [25], [30]. As this kind of genetic programming requires a great deal of time, however, it is best suited to offline evolution.

On the other hand, when online evolution is required, then evolution speed becomes a critical factor. In such cases, direct manipulation of the hardwire structure by GA's is more suitable. In line with our emphasis on online evolution, this will be the focus of our discussions below.

The structure of a reconfigurable hardware device can be changed any number of times by downloading into the device a software bit string called configuration bits. (fField programmable gate arrays (FPGA's) and programmable logic devices (PLD's) are typical examples of reconfigurable hardware devices.

Fig. 1 shows the simplified structure of a commercial PLD called GAL [5]. GAL consists of a fuse array and a logic cell. GAL also has a configuration bit register that determines the link settings to implement a particular hardware structure in the GAL. The fuse array is used to determine the interconnections between the device inputs (e.g., P2 in Fig. 1) and a logic cell, as well as to specify the AND-term inputs to the logic cell. If a link on a particular row of a fuse array is connected (shown as a dot in Fig. 1), then the corresponding input signal is connected to the row. When a link is connected, the corresponding bit in the configuration bit register is set to one. The input signal then becomes one of the AND-term members. For example, the input signals P3 and P5 are connected to the first row, because the corresponding fuse links are connected. These inputs generate the AND-term, P3 $*$ P5. The logic cell can select a predetermined function, such as an AND gate,

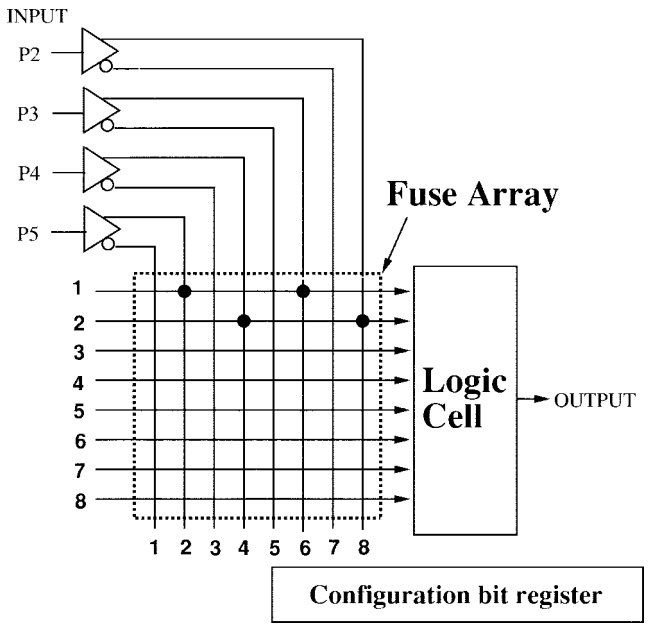


Fig. 1. The structure of PLD.

an OR gate, or a flip-flop, by specifying some bits in the configuration bit register. Thus, according to the content of the configuration bit register, arbitrary hardware functions can be implemented in the GAL.

A GA is a robust search algorithm loosely based on population genetics. It effectively seeks solutions from a vast search space at reasonable computation costs. The basic concept behind the combination of these two elements in EHW is to regard the configuration bits for reconfigurable hardware devices as chromosomes for GA's (see Fig. 2). If a fitness function is properly designed for a task, then the GA's can autonomously find the best hardware configuration in terms of the chromosomes (i.e., configuration bits).

To give a basic idea of how the hardware evolution proceeds, let us take the example of image data compression. For image data compression using EHW, we use a prediction function. Optimal prediction functions vary greatly according to the characteristics of data to be compressed. Some data may include many different shapes and colors, but some may be more plain. It is, therefore, not possible to design in advance a prediction hardware function. Instead of specifying a detailed hardware design, we define a fitness function to allow the EHW to autonomously and adaptively reconfigure itself for the most suitable prediction hardware. For this purpose, the data compression ratio is used as a fitness function. Accordingly, a prediction function circuit with a higher data compression ratio is likely to remain in a population. When a good chromosome is obtained, it is immediately downloaded into the reconfigurable device.

If the prediction performance of a given EHW is reduced due to changes in the nature of the data to be compressed, then the GA process is invoked and the search for a better hardware structure of prediction is initiated. In this way, EHW is capable of both autonomous and dynamic hardware reconfigurations. Below we show how this characteristic of EHW is utilized in five EHW chips and their applications.

III. Gate-Level EHW Applications and Chip Architecture

This section describes two applications using gate-level evolution and the architecture of a general purpose EHW chip based on gate-level evolution.

A. Myoelectric Prosthetic Hand Controller with EHW

1) Background: Although limited to a simple recognition task, Iwata's work in character recognition has demonstrated that EHW is capable of performing the noise-insensitive pattern recognition task as well as in neural networks [9]. This work was inspired by Armstrong's ALN which demonstrated that Boolean logic can be employed to deal with noise-insensitive pattern recognition [14]. Using EHW, more compact implementations and faster processing speeds are possible than with neural networks.

The myoelectric prosthetic hand control is our second application in this direction (our first being the character recognition). The myoelectric hand is operated by the signals generated with muscular movements (electromyography, EMG) [20]. It takes a long time, however, usually almost one month, before a disabled person is able to control a multifunction prosthetic hand freely. During this period, the disabled person has to undertake training to adapt to the myoelectric hand. We have reversed this situation by having the myoelectric hand adapt itself to the disabled person and thus drastically reduce this training period.

Although work is being carried out on applying neural networks for adaptable prosthetic hand controllers, this approach is not very promising due to implementation problems because systems using neural networks are large and thus difficult to implement within a prosthetic hand.

In contrast, the system using the EHW is suitable for this application, because of its compactness and high-speed adaptability. The adaptation speed with EHW is usually less than 10 min, which is a significant improvement as compared with other systems (e.g., one month). It has, therefore, been decided that the EHW-based prosthetic hand will be commercialized.

2) The EHW Learning Method and Its Performance: The myoelectric hand used in this work is able to perform the six actions (Fig. 3), which are paired (open-grasp, supination-pronation, and flexion-extension), with a separate motor control for each pair. The task for the EHW controller is to synthesize pattern recognition hardware to map input patterns (i.e., feature vectors for two-channel EMG signals), to the desired actions of the hand (i.e., one of the six actions).

Because EMG signals vary greatly between individuals, however, it is impossible to design such a recognition circuit in advance. Furthermore, even for a particular person, the feature vectors of the EMG signals sometimes may change even over short periods. Therefore, the control hardware circuit must be synthesized adaptively.

Circuit synthesis is carried out as follows. First, training pattern data is obtained in an online fashion, where the hand user makes one of the six actions from which we measure ten EMG signal patterns. For all six actions, therefore, we obtain a set of 60 signals as training data. Then, based on this obtained training data, circuit synthesis is started using a GA.

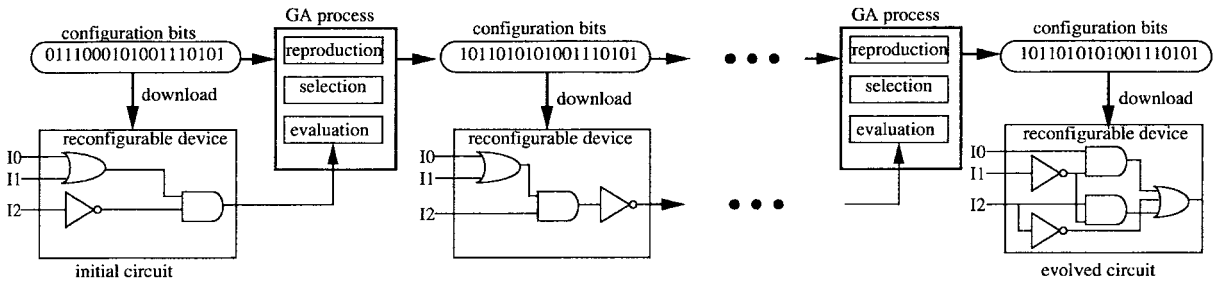


Fig. 2. Basic concept of evolvable hardware.

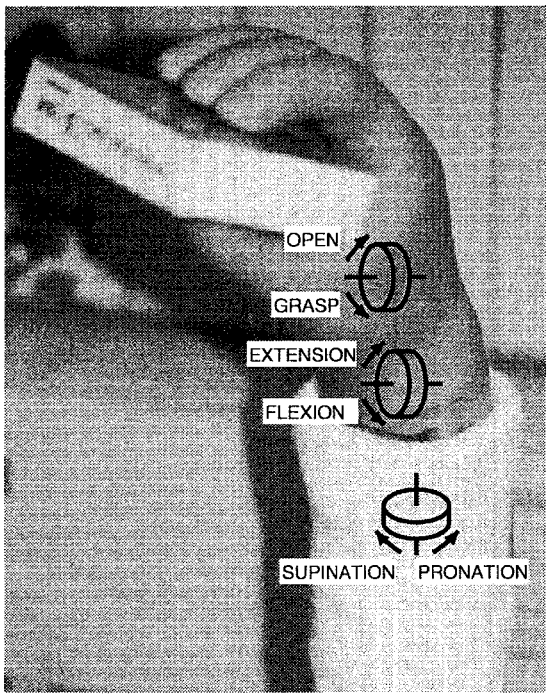


Fig. 3. Myoelectric prosthetic hand.

The GA adaptively implements the circuit on the PLA (programmable logic array, which is a similar device to PLD) in the EHW controller. The number of the product term lines in the PLA is 32. The output and the input width are 6 and 8 bits, respectively.

The chromosome is 1024 bits long, and the population size is 32. The GA algorithm used is called GRGA (gene replacement genetic algorithm), which accelerates adaptation speed. For details of the GRGA, refer to [15].

After the GA is executed for 5 min (the number of GA evaluations is about 17 000), we measure the performance of the recognition circuit just evolved, by letting the user carry out each hand movement for 10 s. During this period, the actual measured EMG signals enter the evolved circuit, and the circuit outputs are compared with the desired hand action in order to obtain a recognition rate. For hand actions with low recognition rates, additional training data are given and the GA is restarted. Table I shows the results of the learning after 5 and 10 min. After 5 min, the average recognition rate over the six actions is 57%. After providing additional training patterns and further 5 min for GA execution, the average recognition rate is 81%.

TABLE I
OUTPUT PATTERN RATES OF SYNTHESIZED CIRCUIT, WHICH CORRESPOND TO EXPECTED OUTPUT PATTERNS (AVERAGED FOR THREE PEOPLE)

	before training pattern addition (%)	after training pattern addtion (%)
SUPINATION	66	74
PRONATION	49	72
FLECXION	67	88
EXTENSION	84	95
GRASP	38	75
OPEN	36	84
AVERAGE	57	81

These results were obtained from experiments using an EHW simulator. When the gate-level EHW chip, as described later, is installed in the hand, the GA execution time is reduced drastically.

B. Self-Adapting Robotic Navigation

The advantages of being able to adapt hardware structure to changing environments with EHW is demonstrated clearly in the navigation task of a mobile robot that tracks a colored ball while avoiding obstacles and that adapts its behavior when sensors break [1].

While robots, such as arm manipulators, are mostly programmed in a very explicit way to execute well-defined tasks, mobile robots are forced to work in environments that are unknown and dynamic in nature. The robot has no *a priori* knowledge of the shapes and positions of obstacles or of the position of the colored ball. The robot also has to acquire knowledge about the configuration of its sensors, which can be blinded or disabled from time to time.

The robot's behavior is based on its current sensory input and its previous interactions with the environment. Its controller can be considered as a kind of adaptive reactive system, which can be described by a dynamic Boolean function easily implemented in hardware using EHW. The robot learns how to navigate in the environment by online evolution and builds an explicit model of the environment as a collection of events. The simultaneous learning of both an implicit model of the environment within the genetic population and an explicit

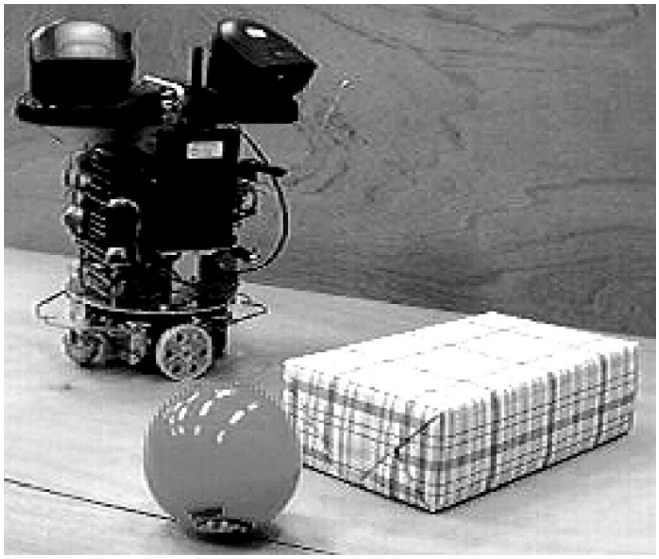


Fig. 4. Autonomous mobile robot.

model can reduce the number of fitness evaluations required in the real physical environment because the fitness evaluations can be carried out using the explicit model. The reduction in the number of required interactions together with the fact that the fitness evaluations are executed using evolvable hardware has the advantage of speeding up the adaptive process.

1) The Robot and Its Environment: The robot shape is circular with a diameter of 25 cm (Fig. 4). It has ten proximity infrared sensors of 1-bit resolution equally distributed around its periphery (Fig. 5). These sensors indicate the presence of objects at distances of less than 30 cm, yielding 10 bits of sensory input to the reactive robot controller. Two cameras are mounted on the top of the robot in order to locate the target within the four 90° sectors around the robot. The sector number locating the target is represented with 2 bits of sensory input. Thus, 12 bits of sensory input (proximity: 10 bits, target direction: 2 bits) are provided to the reactive robot controller which is implemented by one EHW circuit.

The output of the reactive controller is a 3-bit motor command to determine the motion of the robot. The robot moves using two independent motor wheels, which are controlled in terms of speed. This allows the robot to perform eight preprogrammed motions: two translations, two rotations, and four combinations of rotation and translation.

The robot is controlled by a TMS 320 C44 board connected to one transputer for the infrared sensors, the vision sensor, and the motor wheels, and two evolvable-hardware circuits that execute the robot controller and simulate the evolutionary process, respectively.

The robot also has four bumping sensors to detect when it hits an obstacle at either the front or the back. The number of bumps during a navigation run is counted and used in the fitness calculation to evolve the reactive robot controller, where a good navigation controller is one that yields fewer bumps.

2) EHW and Its Performance: The EHW circuit on board the robot implements a dynamic reactive robot controller that accepts 12 bits of sensory inputs and yields 3-bit wheel motor commands. When the robot does not detect obstacles in its neighborhood (its ten proximity sensors are all OFF), its behavior is preprogrammed on the EHW to go toward the ball. When the robot detects obstacles (at least one of the proximity sensors is ON), the EHW implements a dynamic Boolean function with three FPGA Altera Flex 8000's. The Boolean function is implemented in terms of a 50-term disjunctive normal form (DNF) with an AND-array and an OR-array (Fig. 5). In our experimental setup with 12 Boolean input sensors and three Boolean outputs, the AND–OR array has 27 columns and needs a maximum number of 4096 rows to represent any three-output Boolean function of the 12 Boolean variables. To force the Boolean function to generalize, however, the number of rows can be reduced by merging rows with the same output. Our experiments were able to reduce the number of rows and thus the number of terms of the DNF to 50.

The configuration bit string of the 50-term DNF is 1350 bits long while 12 144 bits are needed to represent any function with 12 Boolean inputs and three outputs. The 1350 bits of the EHW are regarded as a chromosome for the GA.

To learn with fewer interactions with the physical environment and still maintain good online performance, the robot has to do some experimentation in an approximate explicit world model of the environment. Our approach consists of having the robot learn a model continually throughout its lifetime and, at each motion, the current model is used to compute an optimal controller that controls the robot until it hits an obstacle or becomes stuck. This method, first, avoids an arbitrary division between the learning phase and motion phase. Second, with this approach the robot is able to gather data about the environment progressively depending on the efficiency of the controller. Third, the robot is able to deal with changes in the environment or in its structure. Because this approach is extremely demanding computationally and to accelerate the entire evolutionary process, the evolutionary process was implemented with evolvable hardware situated next to the evolvable hardware controlling the robot. The evolvable hardware evaluated a population of controllers by computing their Boolean functions. Each controller was evaluated in the explicit world model using an experience replay strategy [13]. The fitness function is a combination of three factors: 1) the number of times the robot hits an obstacle, 2) the number of times the robot became stuck near an obstacle, and 3) the distance covered by the robot in a fixed period of time. In our approach, evolution is used as an adaptive strategy searching continuously for controllers to avoid bumping into the obstacles and becoming stuck near an obstacle, and to cover the greatest distance in an explicit world model that is gradually changing. The population size of the GA was 500 individuals. For the selection algorithm, we used tournament selection with tournament size 20 and an elitist algorithm. Using these three factors and the preprogrammed behavior, the real robot was able to reach the target by searching and switching controllers in the population every time the world model changed.

The performance of this robot navigation system is remarkable, in comparison with other autonomous robots, both in terms of its adaptation speed and the degree of self-adaptation.

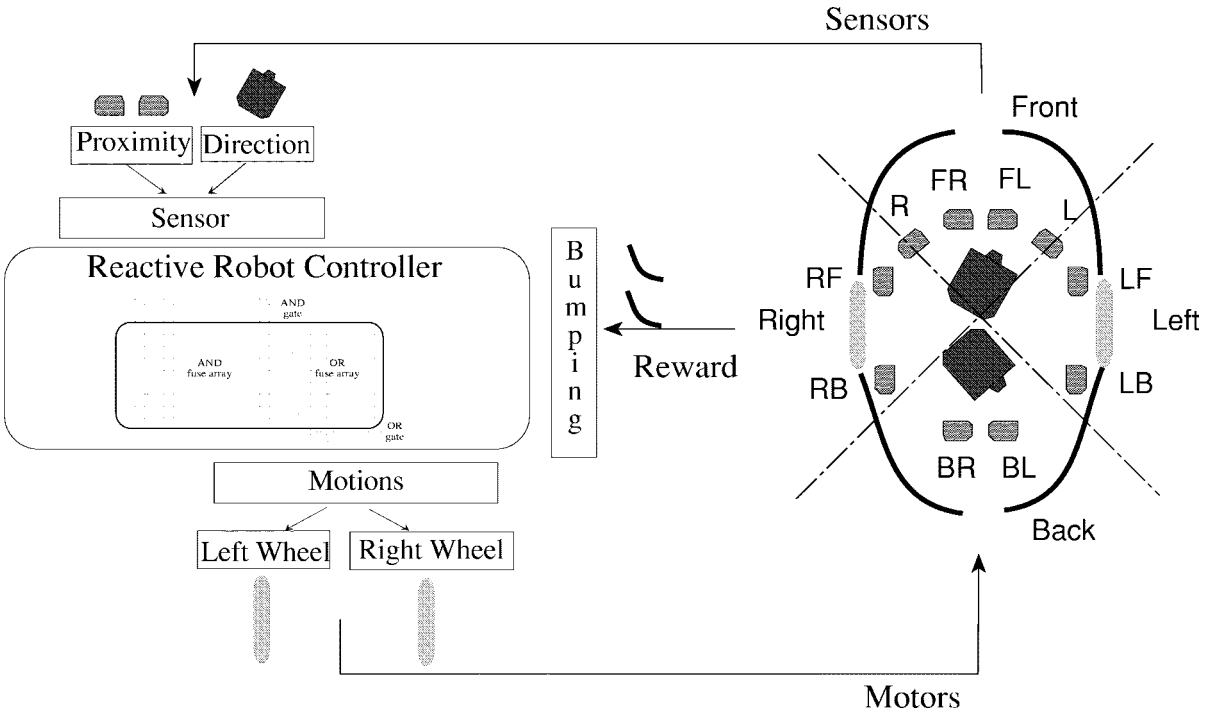


Fig. 5. Reactive navigation system.

In general, within 10 min on average, the robot is able to find a set of suitable hardware configurations for the reactive robot controller and is able to switch dynamically, and in real-time between these hardware configurations to track the ball while avoiding obstacles. This is two orders of magnitude faster than previous work [12].

When some of the proximity sensors are intentionally blinded, the robot will autonomously start to search for another set of hardware configurations for the reactive robot controller and within a few minutes of navigation can succeed in finding a new set of reactive controllers that allow the robot to continue tracking the ball with the remaining functional sensors.

The EHW implementation is almost six times faster than a digital signal processor (DSP) simulation with a TI C44 processor, but the speed can be improved further by employing the EHW chip described in the next section.

C. General-Purpose EHW Chip for Gate-Level Hardware Evolution

This chip was developed in April 1998 to serve as an off-the-shelf device for gate-level hardware evolution (Fig. 6). It was developed for an autonomous mobile robot and a myoelectric artificial hand.

In most research on EHW, GA's are executed with software on personal computers or workstations. This makes it difficult to use EHW in situations that need circuits to be as small and light as possible. For example, a prosthetic hand should be of the same size as a human hand and weigh less than 700 grams. Similar restrictions exist for autonomous mobile robots with EHW controllers. One answer to these problems is to integrate both the GA hardware and the reconfigurable logic on to a single LSI chip.

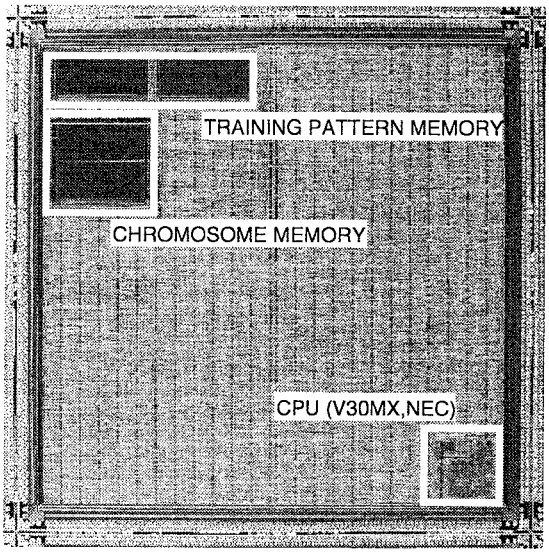


Fig. 6. Gate-level EHW chip.

This has been done with the digital EHW chip, which consists of three components: 1) a PLA, 2) the GA hardware with a 2K word chromosome memory and a 2K word training pattern memory, and 3) a 16-bit 33 MHz CPU core (NEC V30; 8086 compatible). Arbitrary logic circuits can be reconfigured dynamically on the PLA component according to the chromosomes obtained by the GA hardware. The CPU core interfaces with the chip's environment and supports fitness calculations when necessary. The size of the GA hardware, excluding memories, is about 16K gates. In terms of gate size, this is almost one-tenth of a 32-bit CPU core (e.g., NEC V830). However, genetic operations carried out by this chip are 62 times faster than on a Sun Ultra2 (200 MHz).

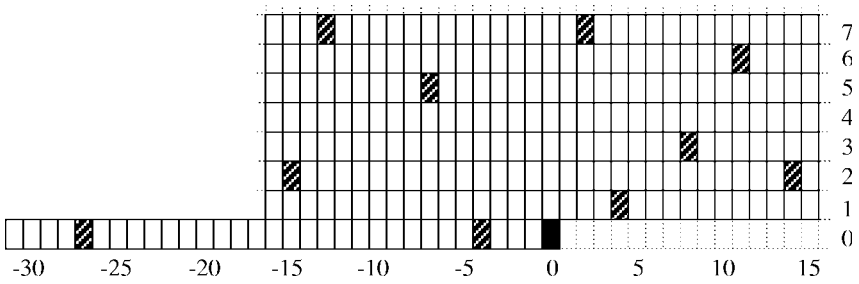


Fig. 7. Template for pixel prediction.

We are currently developing the next version of this chip with lower power consumption and a smaller chip size.

IV. Data Compression Chip for Electrophotographic Printing

Electrophotographic (EP) printing is the latest generation technology in the printing and publishing industry, making it possible to print books with high-precision photo quality. An EHW-based data compression chip has been developed, and it has outperformed current international standards of lossless data compression for bilevel image data. This chip is a gate-level digital EHW dedicated to a particular application.

A. Data Compression of Electrophotographic Images

Data compression devices are essential for the design of EP printers, which handle large amounts of data very quickly. For example, one A4-size EP image of 1200 dpi requires 70 MB for storage, and EP printers process hundreds of different pages at a speed of 100 page/min. (Note that normal color copiers can print less than 10 pages/min.) This means that to print a book with 100 pages, 7 GB of image data must be transferred to the printer at a speed of 1800 MB/min. Unfortunately, the data transfer speed of normal hard disk drives is only 300 MB/min. EP printers, therefore, have to employ data compression techniques 1) to compress image data efficiently and 2) to reconstruct the compressed data very quickly. Moreover, the reconstructed image must be identical to the original to avoid deterioration of quality, such as blurred letters and distortions in images (lossless compression). Traditional lossless data compression techniques, however, are insufficient both in terms of compression ratios and decompression speeds.

The EHW data compression chip can solve these two problems by a precise prediction mechanism using reconfigurable hardware [3]. Image data consists of the values for many pixels. Because the value of each pixel tends to be closely related to the values for neighboring pixels, it is possible to predict the value of a given pixel based on the values for its neighboring pixels. If the value can be predicted correctly, it is not necessary to store it separately, which represents a saving in the size of the image data. This means that compression ratios depend greatly on the precision of the predictions. To increase the compression ratio, it is necessary to continually reselect the most suitable prediction mechanism for the varying patterns within an image.

TABLE II
Comparison of Data Compression Ratio

	Printer Image	Fax Image
Lempel-Ziv	3.34	8.41
JBIG	3.35	14.67
EHW	6.52	19.82

B. Prediction of Pixel Values by EHW

GA's within the EHW can be used to determine which pixels to refer to at prediction. The pattern of locations for such reference pixels is called a "template" [23]. With the original image divided into a number of subimages, GA's search for a set of optimal templates for the subimages. Once a set of optimal templates has been discovered, the hardware prediction mechanism is reconfigured accordingly. This leads not only to improved compression ratios but also to higher decompression speeds, because decompression is also carried out by the EHW hardware.

A template consists of ten pixel locations, and each location is selected from a 32×8 area (Fig. 7). Each location is represented by 8 bits ($2^8 = 32 \times 8$), with 80 bits needed to indicate all ten locations for the template. The chromosome population for the GA's represents templates that are used in data compression, where the size of data after compression is assigned back to the chromosome.

Table II shows a comparison with two major international standards for data compression: Lempel–Ziv ("compress" command of UNIX) and JBIG (joint bilevel image coding experts group) [21], [22], both available on LSI chips. The EHW chip attained compression ratios for printer images almost twice those obtained by the international standards. Arrangements have already been made for this compression mechanism to be used in a commercial EP printer.

C. Architecture of Data Compression EHW Chip

The data compression EHW chip mainly consists of two parts: NEC V830 RISC processor (32-bit, 100 MHz) and the data compressor (Fig. 8). The V830 controls the procedures in data compression, runs the GA calculations, and interfaces with the host computer. The data compressor hardware receives the optimal template identified by the GA, compresses the input image (Fig. 9), and returns the size of compressed data to the V830 for the GA evaluation.

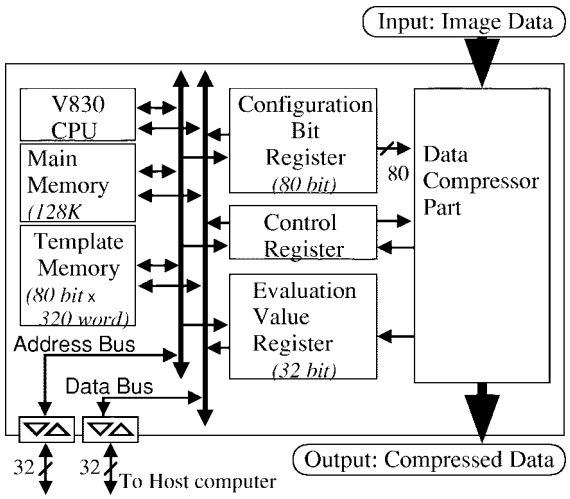


Fig. 8. The organization of the data compression EHW chip.

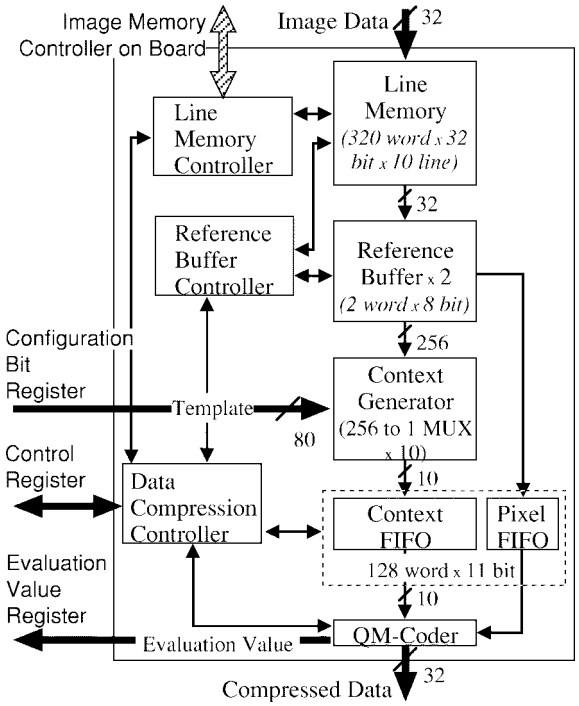


Fig. 9. A block diagram of the data compressor.

The data compressor hardware works at 33 MHz, so the EHW chip can compress and decompress a large EP image within a few minutes. Therefore, this compression chip can satisfy the severe requirements for processing speed of EP printers.

V. The GRD Neural Network Chip

A. Overview

The GRD (genetic reconfiguration of DSP's) chip is a function-level evolvable hardware device designed for neural network applications [7]. The goal of the development is to liberate neural network users from the difficult design task of identifying an optimal neural network. Using GA's, both the topology and the hidden-layer node functions of a neural network mapped on the GRD chips can be dynamically reconfigured. This online learning capability allows neural networks to adapt dynamically to changing problems.

In neural network applications, optimal performance for a given problem is obtained by creating a neural network with the most suitable topology and the most appropriate node functions (e.g., sigmoid function or Gaussian function [19]). Furthermore, to meet the time constraints imposed by real-time applications, neural network hardware systems need to be "tailored" to an ideal network size for a problem. In general, it is very difficult to design an optimal neural network and process it with scalable parallel hardware [17], [18].

With the GRD chip, however, the GA program on the RISC processor can continuously reconfigure the neural network topology and node functions to maintain an optimal performance.

The GRD chip consists of a 32-bit RISC processor and a binary-tree network of 15 DSP's. Each DSP can execute one node function. Using a binary-tree network, multiple GRD chips can be easily connected to configure scalable neural network hardware.

In addition, because the RISC processor is incorporated within the GRD chip, it does not need a host machine to control these tasks. This is desirable for embedded systems in practical industrial applications, together with its fast online learning capability.

Manufacture of the GRD chip began in April 1998. The results of a simulation with an adaptive equalizer in digital mobile communication have shown that execution with a single GRD chip took 2.51 s, whereas execution on a Sun Ultra2 200 MHz takes 36.87 s. Planned uses for the GRD chip include applications that vary over time and have real-time constraints, such as CATV modems.

B. Genetic Learning

The neural network considered here is aimed at industrial applications that need neural networks for the approximation of nonlinear functions.

Genetic learning determines the network topology (e.g., the number of nodes) and the choice of node functions (e.g., Gaussian or sigmoid function) adaptively for a given application. The initial values of the weights and the parameters of the node functions are also determined by the GA and then tuned by local learning with the steepest descent method.

Fig. 10 illustrates this genetic learning. A chromosome for the GA represents one network. The network is evolved by applying genetic operators to the chromosomes. For example, Fig. 10 shows how a network with (a) two hidden layer nodes is evolved to have (b) 15 nodes.

Here, we show how the network obtained by the GA is mapped onto the GRD chips and how they are reconfigured dynamically using the example in Fig. 10. The network structure obtained by the GA is immediately mapped onto the GRD chips. For example, a neural network having a Gaussian function and a sigmoid function, as in Fig. 10(a), can be mapped onto the GRD chip, as in Fig. 10(c).

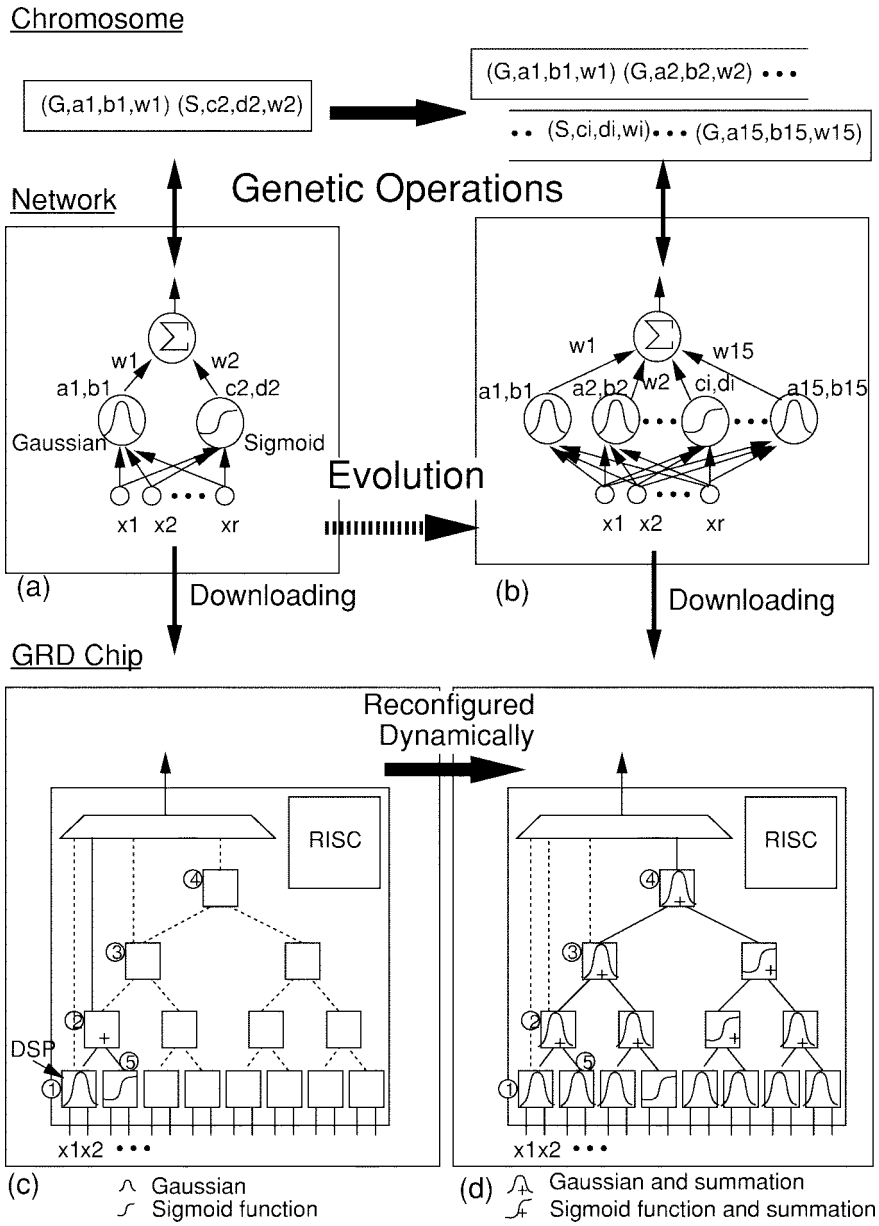


Fig. 10. Genetic learning with the GRD chip.

The functions and tree height in the GRD chips are dynamically controlled by rewriting the chromosomes on the chips. For example, in Fig. 10(c), the output of the GRD chip is connected to the output of DSP no. 2. After evolution, in Fig. 10(d), the output of the GRD chip is reconfigured to be connected to the output of DSP no. 4. Also, in Fig. 10(c), DSP no. 5 calculates a sigmoid function. After evolution, in Fig. 10(d), this DSP is reconfigured to calculate a Gaussian function.

Binary tree connections are very useful when a summation of node outputs is to be calculated. All the DSP's in Fig. 10(d) are configured to conduct the summation in parallel. For example, DSP no. 2 calculates a Gaussian function first and then adds the result to the output from DSP's Nos. 1 and 5.

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

C. The GRD Architecture and the Performance

The GRD chip consists of a 100 MHz 32-bit RISC processor and 15 DSP's. Fig. 11 gives the overall structure of the GRD chip. The RISC processor is a NEC V830 which is designed for multimedia applications. The DSP, a 33 MHz 16-bit fixed-point processor, is called a programmable function unit (PFU). Fifteen PFU's are connected in a reconfigurable network with a binary tree shape. The GRD chip accepts eight 16-bit inputs and generates a 16-bit output with the MIMD parallel processing of the 15 PFU's. The tree shape interconnections are powerful, especially when the summation of neuron outputs is calculated.

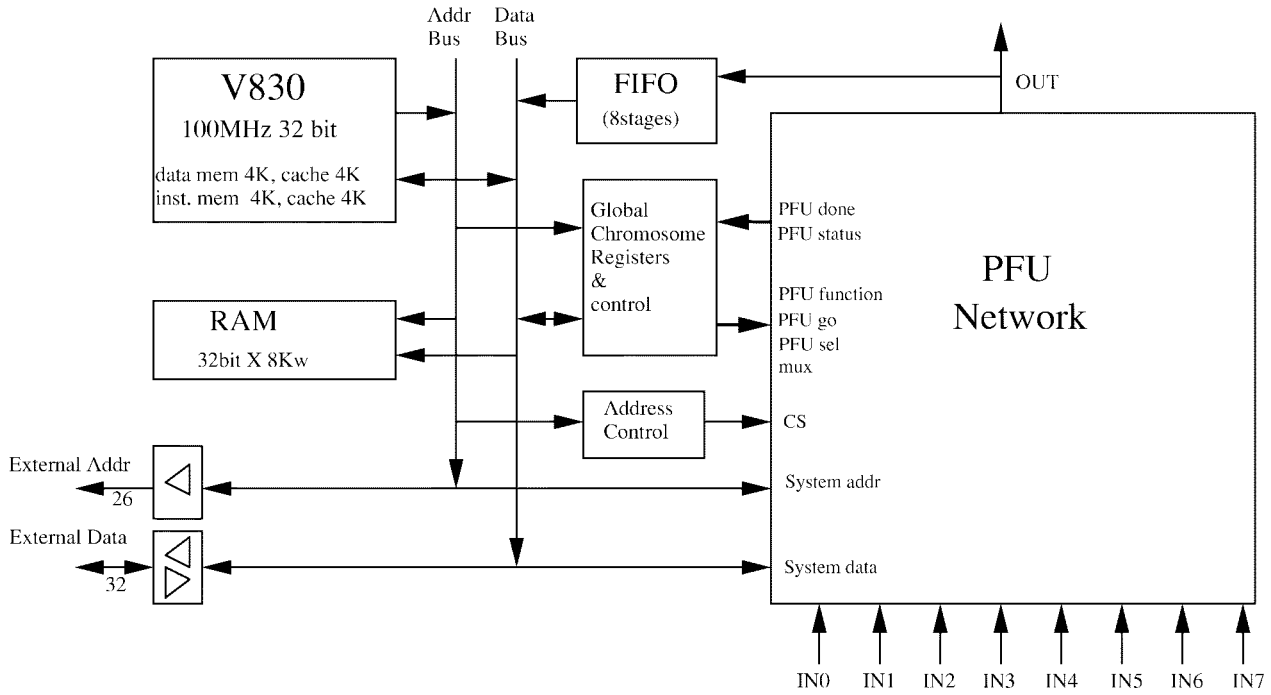


Fig. 11. Overview of the GRD chip.

The GRD chip includes the V830 RISC processor to perform the genetic reconfiguration of the PFU's. This means that the GRD chip can reconfigure itself. In the PFU, a content addressable memory (CAM) is employed to accelerate the computation of nonlinear functions (e.g., sigmoid function and Gaussian). Consequently the GRD chip attains a 319 MCPS (mega connection per second) performance in a multilayer perceptron (MLP).

The GRD chip can process up to 1260 neurons (84 neurons per PFU). To configure a scalable hardware system easily, GRD chips can be connected directly to each other via FIFO buffers inside the PFU. For example, a 19-in rack implementation of 16 VME triple-height boards (nine GRD chips on a board) can achieve the performance of 46 GCPS (giga connection per second) in a MLP.

Simulation results of two applications (chaotic time series prediction and adaptive equalization in digital mobile communication) show that the performance of the GRD chip is almost ten times faster than a Pentium II processor of 400 MHz.

VI. Analog EHW Chip for Cellular Phone

A. Overview

Due to the remarkable advances in recent CPU's and DSP's, applications with analog circuits are rapidly being replaced with digital computing. There are still many applications that require high-speed analog circuits. Communication is one such application.

An inherent problem in implementing analog circuits is that the values of the manufactured analog circuit components, such as resistors and capacitors, will often differ from the precise design specifications. Such discrepancies cause serious problems for high-end analog circuit applications. For example, in intermediate frequency (IF) filters [24], which are widely used in cellular phones, even a 1% discrepancy from the center frequency is unacceptable. It is therefore necessary to carefully examine the analog circuit and to discard those which do not meet the specifications.

The analog EHW chip for IF filters can correct these variations in analog-circuit values by GA's [8]. Using this chip provides us with two advantages. The first is an improved yield rate. If an analog EHW chip is found not to satisfy the specifications, it can be corrected before shipping. This is done by executing the GA in the LSI tester at the factory to alter the defective analog circuit components in line with the specifications.

The second advantage is smaller circuits. One way to increase the precision of component values in analog LSI's has been to use large-valued analog components. This involves larger circuits and accordingly higher manufacturing costs and greater power consumption. With the EHW chip, however, the size of the analog circuits can be made smaller. Obviously, smaller IF filters are particularly welcome in cellular phones, but similar considerations exist in a wide variety of applications where analog circuits are used.

B. The Chip Architecture and Yield Rate

Fig. 12 illustrates the analog EHW chip, which is an integrated G_m-C IF filter. The filter chip is fabricated in a 0.8 μm CMOS process. The active area of the filter is 17 mm^2. The specifications are shown in Fig. 14. The filter has a pass bandwidth of 21.0 kHz centered at 455 kHz, and stop bands specified at attenuations of 48 and 65 dB. Filter gain should be within the dotted lines in the figures. The -3 dB points should be within $455-10.5\pm1$ kHz and $455+10.5\pm1$ kHz. These specifications are very hard to satisfy, because the -3

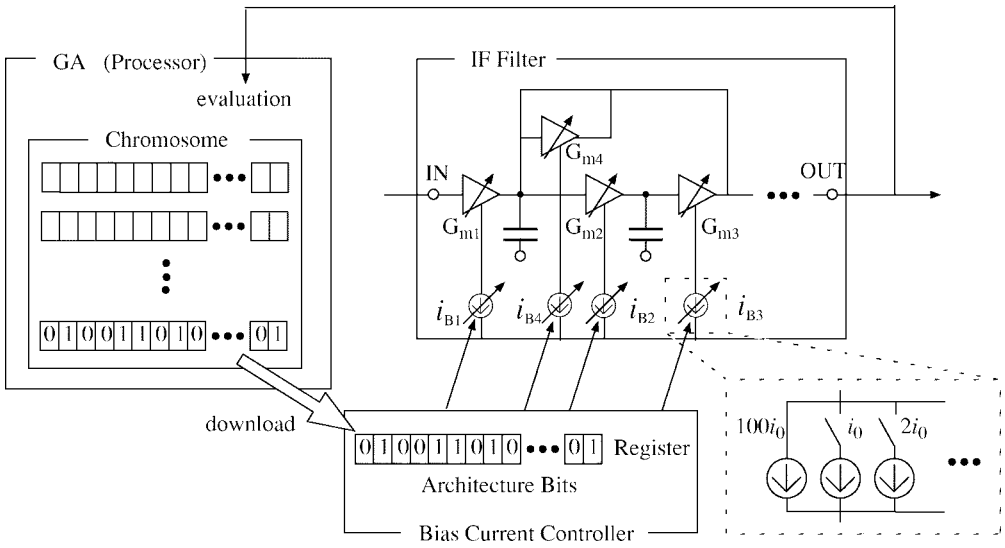


Fig. 12. Adjustment using GA's.

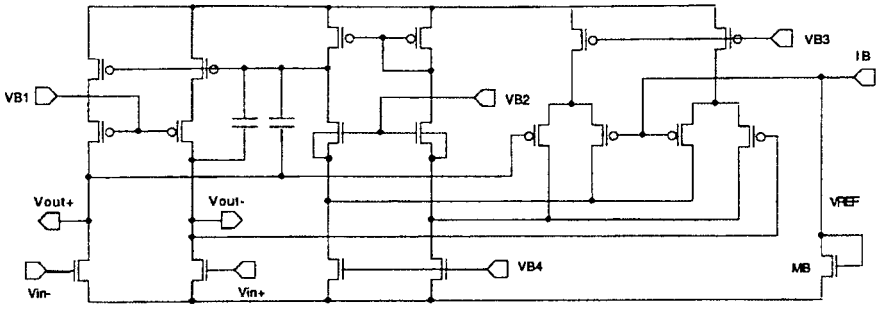


Fig. 13. Schema of G_m amplifier.

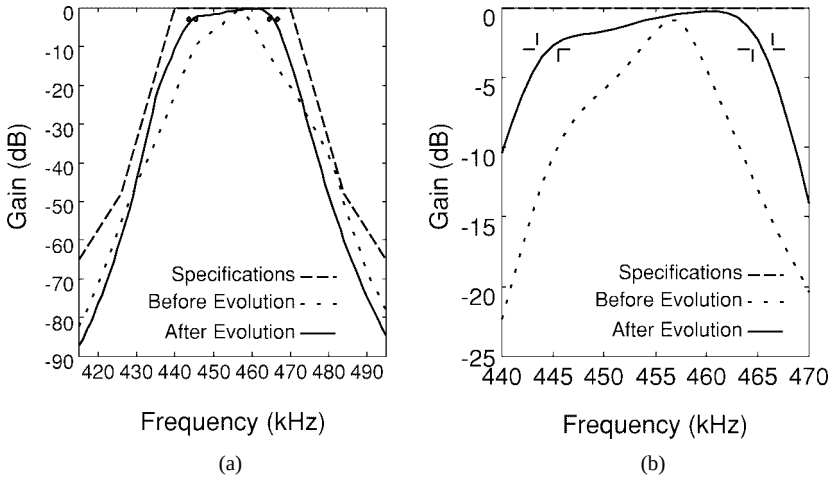


Fig. 14. (a) Frequency responses and (b) magnification of (a).

dB points will be outside these limits if the center frequency is shifted even by 1%.

This IF filter has 39 parameters in total: 16 of these are related to the center frequency, 16 for bandwidth, and three for filter gain. In the integrated circuit of the filter, these parameters correspond to the transconductance of the G_m amplifiers. The schema of a G_m amplifier is shown in Fig. 13. Each transconductance may differ greatly from the target values, however, by up to as much as 20%.

To correct these variations, the value of the G_m amplifier can be set genetically. The values, which actually control the bias currents to the G_m amplifiers, are coded as configuration bits. The GA, which is executed on an external PC, determines the optimal configuration bits.

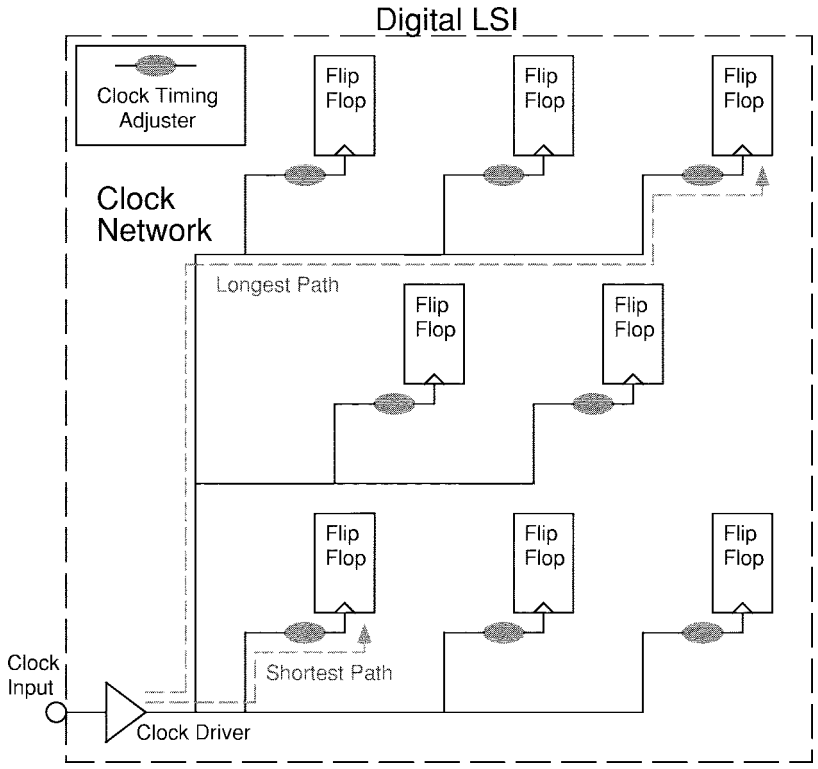


Fig. 15. EHW-based clock-timing adjusting architecture.

A chromosome for the GA consists of 39 genes that correspond to the filter parameters. Each gene has N bits that determine the transconductance. For example, if $N = 2$, there are four transconductance values for selection by the GA's. The genes 00, 01, 10, and 11 mean that the parameter is multiplied by $1.0-2\times D$, $1.0-D$, $1.0+D$, and $1.0+2\times D$, respectively, where D is a constant value. Fitness is the weighted sum of deviations between the ideal gain and the gain obtained by the EHW chip.

In simulations, each transconductance value in the circuit was assumed to vary from the target value by a Gaussian distribution of $\sigma = 5\%$. The parameters N, D were set to 2 and 0.025, respectively. We used a population of 50 individuals, each represented by a chromosome of length 78 bits. A run terminated after the fortieth generation.

Fig. 14 shows the frequency responses for the best chromosome in a run. After iteration, the best chromosome could satisfy the specifications, which the initial population was unable to meet. Out of 100 runs, 95% of the chips conformed to specifications. We have also tested 20 real chips, out of which 18 chips were successfully evolved to meet the specifications. We are currently developing a second version of the chip in which major circuit components are designed to be smaller than in the present chip.

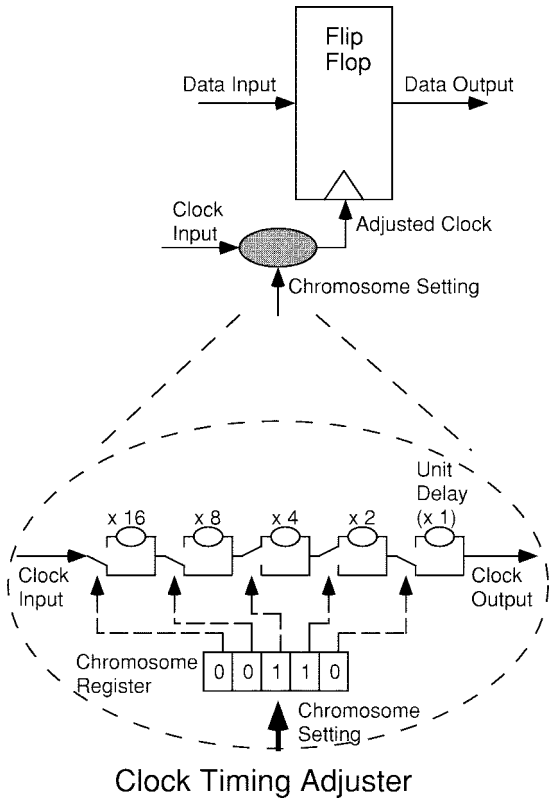


Fig. 16. Clock-timing adjuster.

VII. An EHW-Based Clock Timing Adjusting Chip

A. Overview

The demand for high-speed LSI's, such as Pentium III (500 MHz) and DEC Alpha (600 MHz), is increasing. Unfortunately, the yield rates for such fast digital systems are rather poor. Typically, in the early stages of mass production, yield rates are less than 10%. One of the reasons for the poor yield rates is that the timing delays between digital components

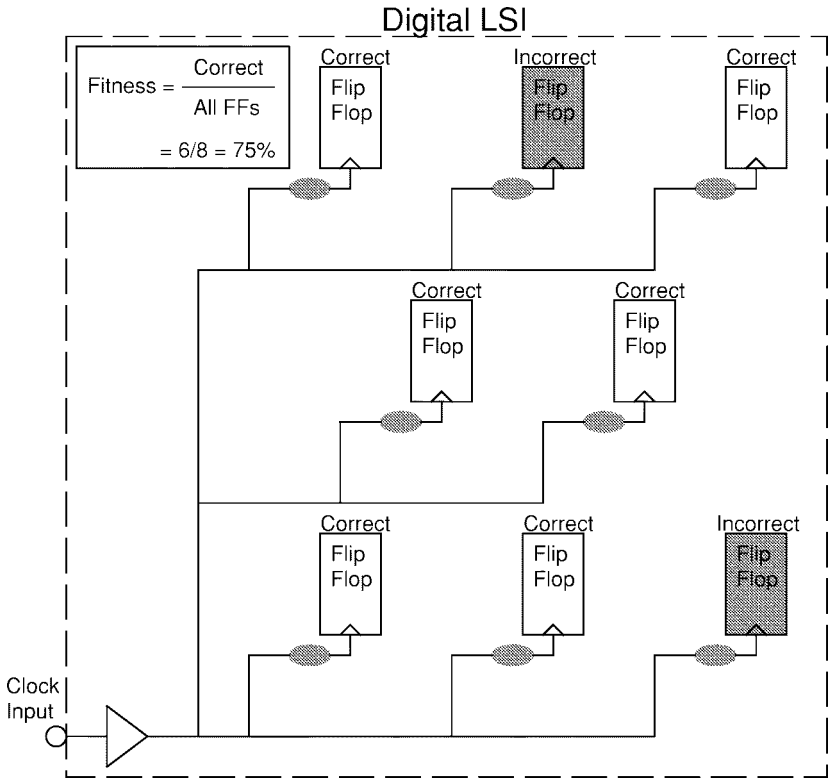


Fig. 17. Evaluation function.

often do not conform to the design specifications [28]. Such discrepancies arise from variations in the values of parasitic capacitances and resistors along the data lines between digital components, which can differ significantly depending on the LSI. Variations in clock timing are referred to as "clock skew." LSI's that fail to satisfy design specifications because of clock skew are simply discarded, leading to poor yield rates.

To solve this problem, we propose an EHW-based clock-timing adjusting architecture for high-speed digital systems [27]. Instead of simply discarding chips that do not meet the specifications, we can genetically adjust the clock timings in the LSI to conform to the specifications.

We have developed a LSI, which is used in the high-speed memory tester, to show the advantages of this architecture. Simulation results show that the number of LSI's that can operate at 800 MHz increases from 2.9 to 51.1% after the clock-timing circuits have been evolved by the GA. This clock-timing adjusting architecture is, therefore, expected to become a basic LSI technology for gigahertz digital systems.

B. The Chip Architecture

The proposed architecture is depicted in Fig. 15. The salient feature of this architecture is the introduction of a delay device, to genetically adjust the clock timing, which can easily be inserted within traditional architectures.

Fig. 16 describes this clock-timing adjuster. It has quite a simple structure, consisting of a chromosome register and a delay generator. The bit-width of the chromosome register depends on the degree of variation in delay time.

The delay generator includes a number of smaller delay devices, with set delays, such as one unit, two units, four units, etc., and these delay devices are connected serially. Each bit of the chromosome register corresponds to one of the small delay devices, and the value of the bit determines whether the delay device is turned on or not. Taking the example depicted in Fig. 16, when the delay has a value of 10 ps, the adjuster would generate a timing delay of 60 ps (10 ps $\times$ 4 + 10 ps $\times$ 2) in total.

C. Genetic Learning

The traditional method for clock-timing adjustment is to correct clock skew. Our architecture is more precise, however, in that it seeks to correct the system to conform to the designer's specifications. Adjustment proceeds as follows.

1) Load a tentative chromosome set into a chip.
2) Conduct an ordinary LSI chip test and supply a series of test data with the system clock running at the specification speed.
3) After the chip test, stop the system clock and extract all FF values.
4) Compare extracted FF values with expected values generated by logical simulation.
5) Calculate the evaluation function from the comparison.
6) Generate a new generation of chromosomes according to the evaluation function.

Note here that the clock timing is not measured directly at all. This is a major advantage because the timing measurement is very costly.

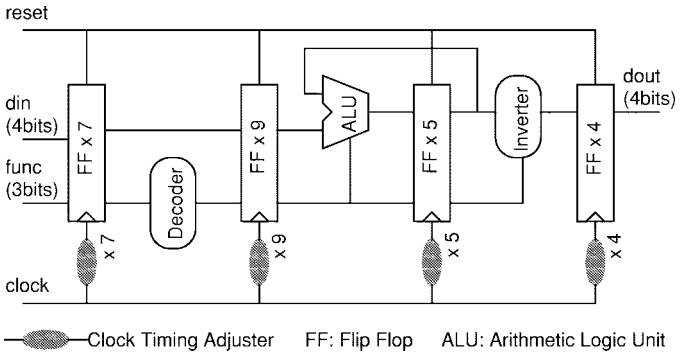


Fig. 18. Evolvable memory test pattern generator.

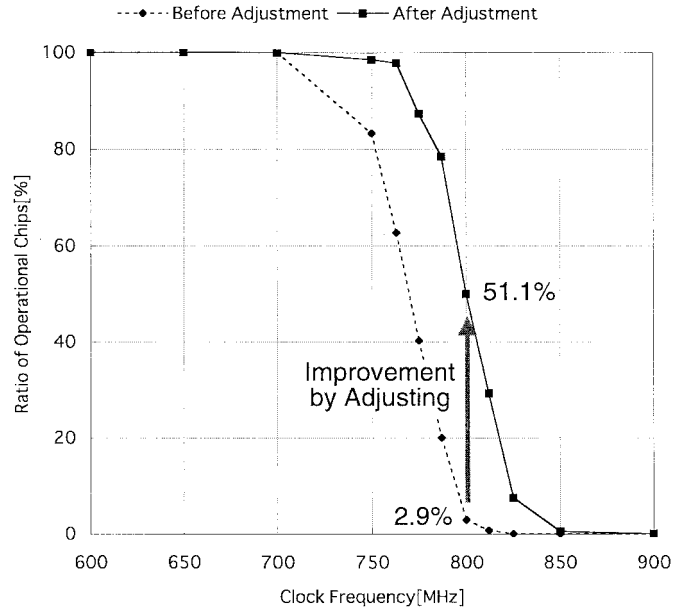


Fig. 19. Simulation result (distribution function).

Fig. 17 shows how to evaluate the fitness. In the figure, two FF's out of eight have failed to work, and the evaluation value is 80%. Chips that do not have a fitness value of 100% are regarded as defective.

D. Simulation Study

The simulation experiments conducted to adjust clock-timing for a circuit are described next.

For the evaluation, a memory test pattern generator, shown in Fig. 18, was selected and converted to a simulation model for evaluation. This is a simple, but quite important, logic block for memory testing devices. It is used to generate access patterns to test memory at speeds used when accessing real memory chips. For example, in testing memory modules for the direct rambus systems, the testing hardware should supply rather long test patterns at a speed of 800 MHz [29].

In the simulation, the number of sample chips was 1000, the population size was 50, and the number of generations was 20. Device parameters were assumed to vary according to a normal distribution. The results are depicted in Fig. 19. The X-axis represents clock speed, with the percentage of chips that could operate at a given speed, being plotted along the Y-axis. The most significant result from these simulations was the finding that 50% of the chips designed with 500 MHz clocks could be adjusted to operate at improved speeds of 800 MHz, whereas only 2.9% could operate at such speeds before adjustment.

The LSI has been developed with bipolar technology, including 2000 gates. A testing environment is now being developed.

VIII. Conclusion

Digital EHW is a key technology for the development of new applications, which previously could not be fully realized due to the lack of autonoumously reconfigurable digital hardware. In general, applications that tend to be time-variant in nature and to have real-time constraints are suitable for EHW, because EHW enables adaptation at the hardware level, and fast execution, being accomplished by the hardware itself.

Although reconfigurable hardware devices, such as FPGA and PLD, are spreading rapidly and the usefulness of reconfigurable hardware is being more widely recognized, reconfiguration in FPGA's is not autonomous and requires human intervention. Thus, EHW indicates a new direction in reconfigurable hardware beyond FPGA's.

After introducing the basic concepts of digital EHW, LSI architectures and their applications, we have described in this paper two analog EHW chip applications.

The analog EHW chip for cellular phones has two advantages: high yield rate and smaller circuit size. These advantages can be applied not only to IF filters but also to a wide variety of analog LSI's. Certainly, we envisage a number of analog LSI's being redesigned according to EHW principles to satisfy severe requirements, especially in high-end analog systems, such as communications and sensoring.

The clock-timing adjusting architecture will be very important for gigahertz digital systems, where although discrepancies from the design specifications are inevitable, they are detrimental to performance. Rather than discarding chips that do not meet the specifications, minute adjustments can, however, be made with EHW architectures and evolutionary computation.

We have discussed the clock architecture within the domain of analog EHW. This is because even digital systems are heavily influenced by the behavior of analog circuits, particularly in the high-frequency range. Without careful consideration of the analog aspects of digital systems, it will be impossible to realize high-end digital systems. In such systems, analog EHW can play an extremely important role.

Acknowledgment

The authors would like to thank Dr. Ohmaki at Electrotechnical Laboratory and Dr. Shimada at RWCP for their support.

References

[1] D. Keymeulen, M. Iwata, Y. Kuniyoshi, and T. Higuchi, "Online evolution for a self-adapting robotics navigation system using evolvable hardware," *Artif. Life*, vol. 4, pp. 359–393, 1999.

[2] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, MA: 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," in *Proc. 15th Nat. Conf. Artificial Intelligence (AAAI-98)*, Madison, WI, July 1998, pp. 486–491.
[4] X. Yao and T. Higuchi, "Promises and challenges of evolvable hardware," *IEEE Trans. Syst., Man, Cybern.t, C,* vol. 29, Feb. 1999.
[5] Lattice Semiconductor Corporation, *GAL Data Book*, 1990.
[6] T. Higuchi, M. Iwata, and W. Liu, Eds., *Proc. 1st Int. Conf. Evolvable Systems* (Lecture Notes in Computer Science, vol. 1259). Berlin, Germany: Springer-Verlag, 1996.
[7] M. Murakawa, S. Yoshizawa, I. Kajitani, X. Yao, N. Kajihara M. Iwata, and T. Higuchi, "The GRD chip: Genetic reconfiguration of DSP's for neural network processing," *IEEE Trans. Comput.*, pp. 628–639, June 1999.
[8] M. Murakawa, T. Yoshizawa, T. Adachi, S. Suzuki, K. Takasuka, D. Keymeulen, and T. Higuchi, "Analogue EHW chip for intermediate frequency filter," in *Proc. Int. Conf. Evolvable Systems*, Sept. 1998, pp. 134–143.
[9] M. Iwata, I. Kajitani, H. Yamada, H. Iba, and T. Higuchi, "A pattern recognition system using evolvable hardware," in *Parallel Problem Solving from Nature*. Berlin, Germany: Springer, 1996, pp. 761–770.
[10] M. Sipper, E. Sanchez, D. Mange, M. Tomassini, A. Perez-Uribe, and A. Stauffer, "Phylogenetic, ontogenetic, and epigenetic view of bio-inspired hardware systems," *IEEE Trans. Evol. Comput.*, vol. 1, pp. 83–97, Apr. 1997.
[11] I. Aleksander, W. V. Thomas, and P. A. Bowden, "WISARD: A radical step forward in image recognition," *Sensor Rev.*, pp. 120–124, July 1984.
[12] D. Floreano and F. Mondada, "Evolution of homing navigation in a real mobile robot," *IEEE Trans. Syst., Man, Cybern., B* vol. 26, pp. 396–407, 1996.
[13] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Mach. Learn.*, vol. 8, no. 3/4, pp. 297–321, 1992.
[14] W. Armstrong and J. Gecsei, "Adaptation algorithms for binary tree network," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-9, no. 5, pp. 276–285, 1979.
[15] I. Kajitani, M. Iwata, H. Yokoi, D. Nishikawa, and T. Higuchi, "An evolvable hardware chip and its application as a multi-function prosthetic hand controller," *AAAI-99*, pp. 182–187, 1999.
[16] J. Holland, *Adaptation in Natural and Artificial Systems.* Ann Arbor, MI: Univ. of Michigan Press, 1975.
[17] E. Fiesler, "Comparative bibliography of ontogenic neural networks," in *Proc. Int. Conf. Artificial Neural Networks.* New York: Springer-Verlag, 1994, pp. 793–796.
[18] Adaptive Solutions Inc., "CNAPs server, preliminary data sheet," Beaverton, OR, 1992.
[19] M. J. D. Powell, "Radial basis functions for multivariable interpolation: A review," in *Algorithms for Approximation*, M. G. Cox, Ed. Oxford, U.K.: Clarendon, 1987, pp. 143–167.
[20] M. Uchida, H. Ide, and S. P. Ninomiya, "Control of a robot arm by myoelectric potential," *J. Robot. Mechatron.*, vol. 5, no. 3, pp. 259–265, 1993.
[21] International Telegraph and Telephone Consultative Committee (CCITT), "Progressive bilevel image compression," Recommendation T.82, 1993.
[22] W. B. Pennebaker, J. L. Mitchell, G. G. Langdon, Jr., and R. B. Arps, "An overview of the basic principles of the Q-coder," *IBM J. Res. Develop.*, vol. 32, no. 6, pp. 717–726, 1988.
[23] S. Forchhammer and K. Jansen, "Data compression of scanned halftone images," *IEEE Trans. Commun.*, vol. 42, pp. 1881–1893, Feb. 1994.
[24] T. Adachi, A. Ishikawa, K. Tomioka, S. Hara, K. Takasuka, H. Hisajima, and A. Barlow, "A low noise integrated AMPS IF filter," in *Proc. IEEE 1994 Custom Integrated Circuits Conference*, 1994, pp. 159–162.
[25] J. R. Koza, F. H. Bennett, D. Andre, M. A. Keane, and F. Dunlap, "Automated synthesis of analog electrical circuits by means of genetic programming," *IEEE Trans. Evol. Comput.*, vol. 1, pp. 109–128, July 1997.
[26] A. Thompson, P. Layzell, and R. S. Zebulum, "Explorations in design space: Unconventional electronics design through artificial evolution," this issue, pp. 167–196.
[27] E. Takahashi, M. Murakawa, K. Toda, and T. Higuchi, "An evolvable-hardware-based clock timing architecture toward giga Hz digital systems," in *Proc. Genetic and Evolutionary Computation Conf.* San Francisco, CA: Morgan Kaufmann, 1999, pp. 1204–1210.
[28] J. Rabeay, *Digital Integrated Circuits.* Englewood Cliffs, NJ: Prentice-Hall, 1996.
[29] Rambus Inc., "Direct rambus technology overview," 1997. [Online.] Available: http://www.rambus.com/docs/drtechov.pdf
[30] T. Hikage, H. Hemmi, and K. Shimohara, "Evolutionary methods for smoother evolution," in *Proc. ICES98*, (Lecture Notes in Computer Science, vol. 1478) 1998, pp. 115–124.

Tetsuya Higuchi received the B.E., M.E., and Ph.D. degrees in electrical engineering from Keio University, Japan.

He heads the Evolvable Systems Laboratory in Electrotechnical Laboratory, AIST, MITI, Tsukuba, Japan. His current interests include evolvable hardware systems, parallel processing architecture in artificial intelligence, and adaptive systems. He is also in charge of the adaptive devices group in the MITI National Project, Real World Computing Project.

Dr. Higuchi received the Ichimura Award in 1994 and the ICES Best Paper Award in 1998. He is a member of the Japanese Society for Artificial Intelligence (JSAI) and of the Institute of Electronics, Information and Communication Engineers (IEICE).

Masaya Iwata received the B.E., M.E., and Ph.D. degrees in applied physics from Osaka University, Japan, in 1988, 1990, and 1993, respectively.

He was a postdoctoral fellow at ONERA-CERT, Toulouse, France in 1993. He is currently a Senior Researcher at the Electrotechnical Laboratory, AIST, MITI, Tsukuba, Japan. His research interests are in genetic algorithms.

Dr. Iwata is a member of the Institute of Electronics, Information and Communication Engineers (IEICE) and the Information Processing Society of Japan (IPSJ).

Didier Keymeulen received the M.Sc. and Ph.D. degrees in computer science from the AI-Lab of the Vrije Universiteit Brussel, Belgium.

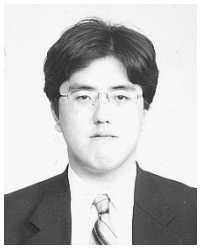
He is a Senior Researcher at the National Electrotechnical Laboratory at Tsukuba, Japan, and also member of the Jet Propulsion Laboratory of the California Institute of Technology, Pasadena. His interests are in bio-inspired complex dynamical systems applied to the design of adaptive embedded systems. In 1996, he joined the Electrotechnical Laboratory as Senior Researcher and started his work on the applications of evolvable hardware for adaptive robotics. He currently develops and infuses the evolvable hardware technology for the NASA future flexible space crafts at the Jet Propulsion Laboratory of the California Institute of Technology.

Dr. Keymeulen was the Belgium laureate of the Japanese JSPS Postdoctoral Fellowship for Foreign Researchers in 1995.

Hidenori Sakanashi received the B.Eng. degree in 1992, the M.Eng. degree in 1994, and the Ph.D. degree for his research on evolutionary computation in 1996, all from Hokkaido University, Japan.

From 1996 to 1998, he was a Fellow Researcher of Japanese Society for the Promotion of Science (JSPS). In 1998, he joined Evolvable Systems Laboratory at Electrotechnical Laboratory, MITI, Tsukuba, Japan. His research interest covers the areas of evolutionary computation, EHW, data compression, autonomous systems.

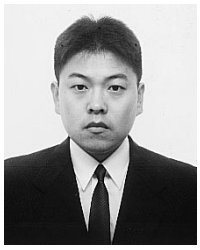
Dr. Sakanashi is a member of the Information Processing Society of Japan.



Masahiro Murakawa received the B.E., M.E., and Ph.D. degrees in mechano-informatics engineering from University of Tokyo, Tokyo, Japan, 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.

Dr. Murakawa 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).



Isamu Kajitani received the B.E., M.E., and Ph.D. degrees in engineering from the University of Tsukuba, Japan, in 1994, 1996, and 1999, respectively.

He is currently working at the Electrotechnical Laboratory, Tsukuba, Japan, as the proposal based Researcher of the new energy and industrial technology development organization. His research interests are engineering applications of genetic algorithms and biomedical engineering.

Dr. Kajitani received the Best Student Paper Award at the Second International Conference on Evolvable Systems. He is a member of the Japanese Society for Artificial Intelligence (JSAI).

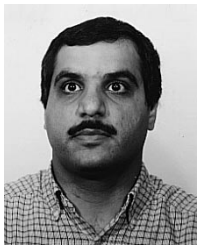
Eiichi Takahashi (M'96) received the B.E. degree in electronic engineering, and the M.E. and Ph.D. degrees in information engineering in 1987, 1989 and 1993, respectively, all from the University of Tokyo, Japan.

He joined the Electrotechnical Laboratory, Tsukuba, Japan, in 1993, where he is currently a Chief Researcher. His research interests lie in computer architecture. In particular, he is interested in evolvable hardware approaches for improving computer architecture and in real-time parallel machines.

Dr. Takahashi is a member the Information Processing Society of Japan, and the Institute of Electronics, Information and Communication Engineers.

Kenji Toda received the B.S. and M.S. degrees in computer science from Keio University, Japan, in 1980 and 1982, respectively.

Since 1982, he has been on the research staff of the Computer Science Division at the Electrotechnical Laboratory, Tsukuba, Japan, where he is currently Leader of the Real-Time Systems Laboratory. His research interests are in the area of parallel processor architectures, interconnection networks, and programming on multiprocessor environment, in particular, on real-time systems.



Mehrdad Salami was born in 1966 in Iran. He received the B.S. degree in electrical engineering from Tehran University, Tehran, Iran, and the M.S. degree in electrical engineering from Sharif University, Iran.

From 1993–1995, he worked on hardware implementation of genetic algorithm at Victoria University, Melbourne, Australia. From 1996 to 1998, he worked as an Industrial Researcher at the Electrotechnical Lab, Tsukuba, Japan, researching on the application of evolvable hardware in image compression. He is currently working on the fast strategies for evolvable hardware in the center for intelligent systems in Swinburne University of Technology, Melbourne. His interests include hardware genetic algorithms, application of evolvable hardware, and fast strategies for evolvable hardware.

Nobuki Kajihara received the B.E. degree in electronics engineering from Yamaguchi University, Japan, in 1981 and the M.E. degree in control engineering from Osaka University, Japan, in 1983.

He is a Principal Researcher at NEC C and C Media Research Laboratories. He joined NEC Corp. in 1986 and has been engaged in the research and development of Parallel Circuit simulation machine, parallel neural network simulation machine, and reconfigurable architecture. His research interests include reconfigurable computing, parallel architectures, and neural networks.

Dr. Kajihara is a member of the Information Processing Society of Japan, the Institute of Electronics, Information, and Communication Engineers, the Japan Society for Artificial Intelligence, and the Japan Neural Network Society.

Nobuyuki Otsu received the B.S., M.Eng., and Dr.Eng. degrees in mathematical engineering from the University of Tokyo, Tokyo, Japan, in 1969, 1971, and 1981, respectively.

Since he joined Electrotechnical Laboratory (ETL), Tsukuba, Japan, in 1971, he has been engaged in theoretical research on pattern recognition, multivariable data analysis, and applications to image recognition in particular. From 1982 to 1983, he was invited to NRC Canada as a Research Scientist. After taking positions of Head of Mathematical Informatics Section since 1985 and of ETL Chief Senior Scientist since 1990, he is currently Director of Machine Understanding Division in ETL since 1991, and concurrently a Professor of the post graduate school of Tsukuba University. He has been involved in the planning of the MITI's ten-year national project, "Real World Computing program" which started in 1992, and is directing the R&D of the project as Head of Real World Intelligence Research Center at ETL.

Dr. Otsu is a member of the Behaviormetric Society of Japan and the Institute of Electronics, Information, and Communication Engineers.